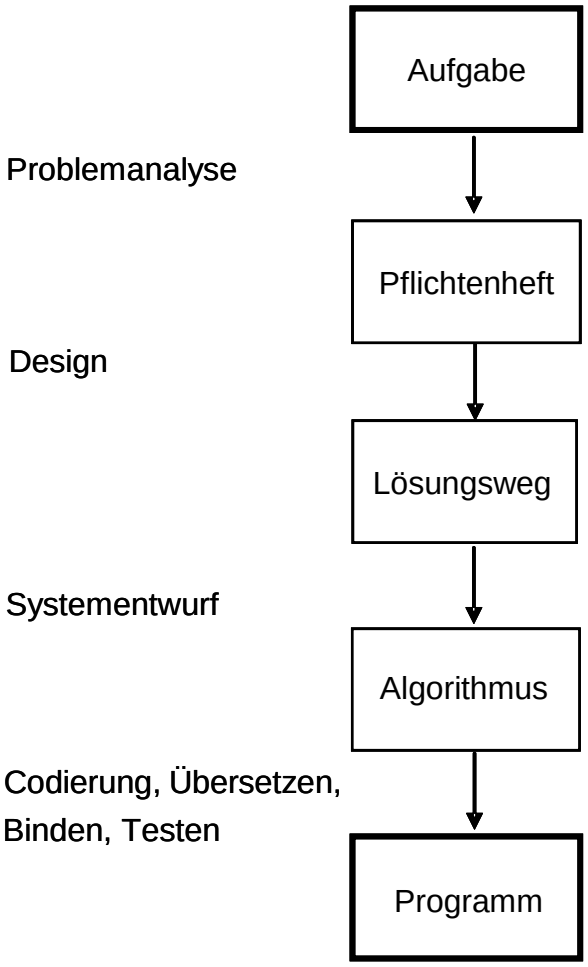
VORLESUNGSSKRIPT I

STRUKTURMECHANIK

für

Studierende der Fachrichtungen Maschinenbau und Mechatronik



Prof.- Dr.-Ing. A. Matzenmiller

Institut für Mechanik im Fachbereich 15 / Maschinenbau
der Universität Kassel
Mönchebergstr. 7
34125 Kassel

WS 2017 / 2018

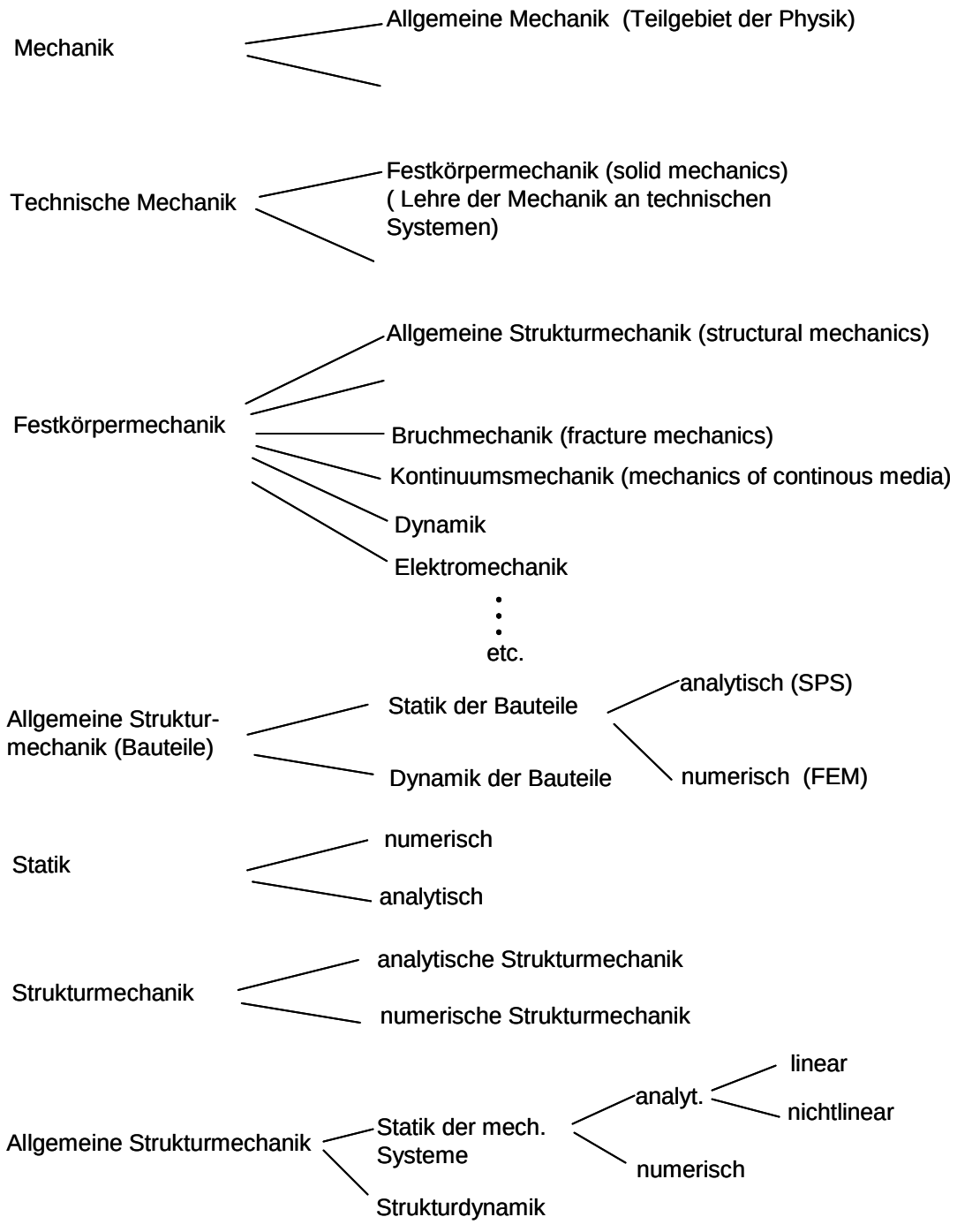
Inhaltsverzeichnis

1 Einführung in die Thematik

1.1 Einteilung der Mechanik und im speziellen der Strukturmechanik

Mechanik ist die Lehre von der Bewegung und den Verformungen von materiellen Körpern (Festkörper, Flüssigkeiten, Gase) unter dem Einfluss von Lasten. Sie ist das älteste Wissenschaftsgebiet der Physik.

Einteilung der Mechanik



Die allgemeine Strukturmechanik kann inhaltlich unterteilt werden in:

- Stabtheorie (Bögen, Rahmen, Fachwerke)
- Ebene Systeme (Platten, Scheiben, Faltwerke)
- Gekrümmte Flächentragwerke (Schalen, Membrane, Halbmembrantheorie),
  z. B. Fahrzeugkarosserie, Flugzeugrumpf, Turbinenschaufeln
- Kontaktmechanik (unilaterale Probleme)
- Schwingungslehre
- Rotordynamik
- Kreiseltheorie
- Robotik

  ⋮

  etc.

Strukturelemente im Sinne der Mechanik sind vereinfachte Bauteile, z.B. Balken, Fachwerkstäbe, Platten. Schalen etc.

eindimensionale Bauteile:	Stab	(Dehnstab, Biegestab, Torsionsstab)
	Seil	(übernimmt nur Zugkräfte)
	Bogen	(gekrümmter Balken)
zweidimensionale Bauteile:	Scheibe	(in der Strukturebene belastet)
	Platte	(quer zur Strukturebene belastet)
	Membran	(nur Zugkräfte)
	Faltwerk	(Scheibe und Platte)
	Schale	(gekrümmtes Bauteil, das in einer Richtung vergleichsweise dünn gegenüber den beiden anderen Richtungen ist)
	Membran- und Biegeschale	
	Axialsymmetrische Strukturen (2-dimensional im Sinne der Mathematik)	
dreidimensionale Bauteile:	(allgemeines Bauteil, dessen Abmessungen in der Länge, Breite und Höhe in derselben Größenordnung sind und beliebig belastet sein kann). Idealisierende Annahmen werden getroffen, was Lagerung, Form (keine Fassung) und Belastung (z. B. Punkt-, Flächen- oder Gewichtlasten) angeht.	

1.2 Ziel der Veranstaltung

- Computer-Programm zur Berechnung von Stäben, Platten und Scheiben verstehen und anwenden können.
- effiziente Berechnung der Spannungs- und Deformationszustände in Bauteilen.
- Beanspruchungszuständen (Spannungen, Dehnungen) in Bauteilen (fachübergreifend) infolge äußerer Belastung (Kräfte, Momente und Temperaturänderung) beurteilen können.
- Entwicklung von Theorien (Dehnstabtheorie, Plattentheorie, Balkentheorie (nach EULER-, BERNOULLI, TIMOSHENKO) aus physikalischen Ideen (kinematisches Verhalten, ebener Spannungszustand, Axialsymmetrie) zur Bauteilberechnung nachvollziehen können.
- Mathematische Formulierung der physikalischen Ideen kennenlernen.
- Einfache Änderung eines linearen Stabwerks- und Plattenprogramms in FORTRAN ausführen können.

2 FORTRAN-Programmierung

2.1 Numerische Datenverarbeitung, Strukturierte Programmierung und Programmsprachen

Zur Lösung von Differentialgleichungen mit Hilfe numerischer Verfahren – wie z.B. die FE - Methode – werden im Lösungsteil (SOLVER) eines FE - Programms fast ausschließlich numerische Operationen verwendet. Für die **numerische** Datenverarbeitung sind in den 50er und 60er-Jahren die höheren Programmiersprachen FORTRAN bei der Firma IBM und ALGOL von der GAMM (Gesellschaft für angewandte Mathematik und Mechanik) und von der ACM (American Society of Mathematics, USA) in Ergänzung zu den maschinenorientierten Assemblersprachen entwickelt worden. Marktwirtschaftliche Gesichtspunkte, die laufende Standardisierung, Abwärtskompatibilität und die Weiterentwicklung einer Programmiersprache sowie die technischen Erfordernisse der FE-Programmierung (numerische Datenverarbeitung, Programmpflege, nicht bezahlbare Kosten für die Umstellung eines kommerziellen FORTRAN–Programms mit i. d. R. mehr als einer halben Million Anweisungen in eine andere Programmiersprache -„never change a winning team“) ließen die FE-Entwickler an FORTRAN festhalten.

Der Hauptanteil der Softwarekosten ist nicht die Programmerstellung anhand der Algorithmen, sondern die Wartung, welche Fehlerelimination, Modifikation und Erweiterung umfasst. Unter einem „Algorithmus[1] ist ein allgemeines, bis in die Einzelheiten festgelegtes, endliches Verfahren“ zu verstehen, „welches zu vorgegebenen Eingangsgrößen Ausgangsgrößen berechnet“ (aus [1]). Gute Lesbarkeit, Wartungsfreundlichkeit und Portabilität eines Rechenprogramms minimieren die Softwarekosten am nachhaltigsten.

Die Lösung einer Aufgabe in der Technik und speziell der Mechanik mit Hilfe eines Computers besteht

- aus der Problemanalyse, d.h. aus der Beschäftigung mit dem „um was es sich dreht“ (z.B. Berechnung der Unbekannten aus einem System von Gleichungen)
- aus dem design, d.h. aus der Formalisierung des Lösungswegs für die gestellte Aufgabe auf dem Papier (z.B. Gausssches Verfahren für die Lösung eines Systems linearer Gleichungen)
- aus dem Algorithmus zum Erreichen des Ziels, d.h. aus einer Reihe von Vorschriften, bestehend aus nicht weiter erklärten „elementaren Aktionen“ (z.B. Grundrechenarten für die Bearbeitung der Zeilen während der Lösung eines linearen Gleichungssystems)
- aus der Codierung des Algorithmus, d.h. aus der Niederschrift aller Anweisungen des Algorithmus` in einer Programmiersprache, wofür alle gängigen Programmiersprachen grundsätzlich geeignet sind – für die kommerzielle FE-Programmierung jedoch Fortran verwendet wird.

In allen Datenverarbeitungsprogrammen werden zur Codierung die drei Grundelemente [2]

Sequenz (sequence): Folge nacheinander abzuarbeitender Anweisungen,

Schleife (loop): Wiederholung eines Programmteils mit unterschiedlichen Werten,

Verzweigung (branching): Ist eine vorgegebene Bedingung erfüllt, so wird ein bestimmter Programmteil ausgeführt, während ein anderer ausgelassen wird.

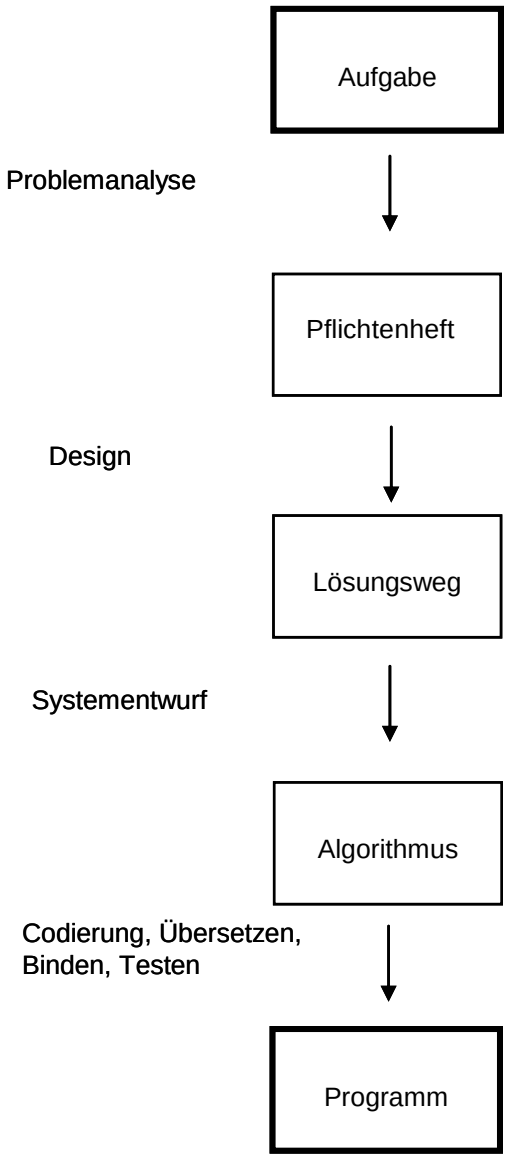
benutzt. Aus diesen drei Basiselementen ist die gesamte strukturierte Programmierung aufgebaut.

Die Hauptaufgabe des Programmierers ist es, eine geeignete Problemlösung in Form dieser drei Bausteine zu beschreiben und diese in äquivalente Anweisungsfolgen in einer Programmiersprache umzuwandeln.

---

[1] Der hochangesehene mittelasiatische Mathematiker und Gelehrte AL CHWARIZMI (gest. etwa 850 n. Chr.) nannte sich lateinisch Algorismi. Davon stammt das Wort Algorithmus.

Lösungsablauf einer Technikaufgabe durch Berechnung mit einem Computerprogramm:



Die sogenannten "höheren" Programmiersprachen sind maschinenunabhängig und heißen "problemorientierte Programmiersprachen". Deren wichtigste Vertreter sind bzw. waren:

BASIC	→	interpretative Sprache, d.h. sie arbeitet interaktiv; braucht daher sehr viel Speicher und ist deshalb für sehr rechenintensive Aufgaben ungeeignet;
		ist einfach zu erlernen; ist jedoch nicht standardisiert
APL	→	unüblich im Ingenieurwesen
ALGOL	→	in der Mathematik (60er Jahre) entwickelt aber nicht vom Ingenieurwesen übernommen worden, das an FORTRAN festhielt

PASCAL → von keiner namhaften Software im Berechnungswesen benutzt;

vor allem in der Mathematik eingesetzt

C → ursprünglich für Systemprogrammierung, also für nichtnumerische Datenverarbeitung entwickelt. C-Compiler werden von Hardware-Herstellern kostenlos mitgeliefert und sind von den Anwendungs-programmierern mißbräuchlich verwendet worden.

C++ → Stärken werden vor allem für objektorientierte Programmierung und Grafikprogramme ausgenutzt

JAVA → Vorteile bei der Erstellung von Computerprogrammen für Internet und Multimediaanwendungen

FORTRAN → Abkürzung für: FORmula TRANslation

→ vorallem von Ingenieuren verwendet

→ alle Löser in großen FE-Programmen sind in FORTRAN geschrieben, da ständige Weiterentwicklung erforderlich ist

→ nicht schlechter als andere Programmiersprachen für die Erfordernisse der FE-Progammierung, also der numerischen Datenverarbeitung, wo kaum Zugriff auf das Betriebssystem notwendig ist

→ Entwicklung hat Mitte der 50er Jahre begonnen,

Mitte der 60er Jahre standardisiert in FORTRAN 66,

stetige Weiterentwicklung bis heute

→ abwärts kompatibel

1962: FORTRAN IV (Industriestandard vorallem für IBM-Rechner)

1966: FORTRAN 66 (Sprachstandard)

1978: FORTRAN 77 (Sprachstandard)

1991: FORTRAN 90 (Sprachstandard)

1997: FORTRAN 95 (Sprachstandard)

2004: FORTRAN 2003 (Sprachstandard)

→ Vorteil gegenüber der Sprache C für FE-Programmierung: restriktiver Aufbau von FORTRAN zwingt zur strukturierten Programmierung und erlaubt daher große Transparenz des Programms, was für die Programmpflege wichtig ist (z. B. keine unübersichtlichen Datenstrukturen wie „Pointer“ auf Felder, deren Elemente wiederum „Pointer“ auf andere Felder sind u.s.w.)

Der Quelltext eines FORTRAN-Programms wird wie derjenige in vielen anderen problemorientierten Programmiersprachen mit Hilfe eines "**Compilers**" (Übersetzer) in Maschinensprache übersetzt und dann mit Unterprogrammen aus Bibliotheken oder anderen Programmteilen mit Hilfe des **“Linkers“** (Binders) gebunden und geladen, so dass ein ausführbares Programm entsteht. Die meisten FORTRAN-Systeme lassen die Einbindung von anderssprachigen Prozeduren (z.B. in C oder Assembler) zu, über die Zugriffe auf Systemressourcen möglich sind.

Hinweis:

Es wird kein FORTRAN-Kurs gegeben, sondern nur die wichtigsten Operationen zum Verständnis eines einfachen FE-Programms in FORTRAN werden vorgestellt. Für die Modifikation eines FE-Programms zur Lösung mechanischer Aufgabenstellungen sind nur wenige FORTRAN-Befehle notwendig. Hierfür wird auf die Schrifttumsstellen [3] [4] und [5] verwiesen.

2.2 Übersicht über FORTRAN - Anweisungen

Ein FORTRAN-Programm besteht aus:

- ausführbaren Anweisungen, d.h. vom Rechner auszuführende Operationen, wie

  z.B. arithmetische Ausdrücke: A=B+C
  Zuweisungen: v = e
  Steueranweisungen: DO
  CONTINUE
  GO TO u.a.
  Ein-/Ausgabeanweisungen WRITE
  READ
  OPEN u.a.

- nicht ausführbare Anweisungen, d.h. Anweisungen an den Übersetzer (Compiler), wie

  z.B. Feldvereinbarungen DIMENSION SUM(10),
  Programmende END
  Unterprogramm FUNCTION
  Unterprogramm SUBROUTINE
  oder weitere Vereinbarungsanweisungen, wie
  z.B. DATA
  EQUIVALENCE
  COMMON

2.2.1 Sprachelemente

Eine FORTRAN-Programmierzeile besteht aus 4 Teilen, die jeweils im Bereich dafür reservierter Spalten stehen müssen:

Spalte 1 - 5: Anweisungsmarke (ein- bis fünfstellige positive Zahl)
6 Fortsetzungszeile (beliebiges Zeichen außer Null oder Blank)
7 - 72: Anweisungsteil
73 - 80: Kommentarteil

Sprachelemente können sein:

i) Namen: maximal 6 alphanumerische Zeichen,
z.B. für Unterprogrammnamen, z.B.: TEMPO
für Schlüsselwörter, z.B.: SUBROUTINE
PRINT
für Formelfunktionen, z.B.: SIN

ii) Konstante: ist ein Datenelement mit festem Wert

Es kann sein:	ganzzahlig	INTEGER	z.B..	237
			oder:	-74
		Wertebereich	-2^{31} bis $+2^{31}$	
	reell	REAL	z.B.	7,5
			oder:	42.E+3
		7 signifikante Dezimalstellen		
	doppelt genau	DOUBLE PRECISION	z.B.	5.83 D-2
		15 - 16 signifikante Dezimalstellen		
	komplexe Zahlen	COMPLEX		
	logische Konstante:	.TRUE.		
		.FALSE.		
	Zeichenkonstante:	z.B. 's'		

iii) Variable (max. 6 Buchstaben):

ganzzahlig:	Konvention ist:	Anfangsbuchstabe I - N	
	z.B.	I	
		NSUM	
reell:	alle anderen Anfangsbuchstaben,		
	z.B.	SUM	
		TEMP	
		X	
doppelt genau:	vereinbaren durch z.B:		
	IMPLICIT DOUBLE PRECISION (A)		
	Alle Variablen mit dem Anfangsbuchstaben A sind doppelt genau.		
	oder:		
	REAL*8		
Komplex	COMPLEX		
Logisch	LOGICAL		

iv) Felder: Datenstruktur mit mehreren Elementen müssen immer vereinbart werden durch DIMENSION oder COMMON

z.B. : DIMENSION VARI (4,3,6)

VARI ist ein 3-dimensionales Feld mit 4 Zeilen, 3 Spalten und 6 Ebenen

v) Zeichenteilfolgen: müssen vereinbart werden durch die Anweisung: CHARACTER

wie z.B. : CHARACTER *6 S1, S2

DATA S1/'STRING'/

S2 = S1

2.2.2 Ergibtanweisungen in FORTRAN

i) Arithmetische Ausdrücke

		Rangordnung (Prioritätenfolge)
Addition bzw. Subtraktion:	+ bzw. -	1
Multiplikation bzw. Division	* bzw. /	2
Vorzeichen (- Minuszeichen)	-	3
Potenzieren	**	4

z.B. 2**3-4 = 2^3 - 4 = 8 - 4 = 4

Klammern: (...) Abarbeiten wie in der Mathematik

Ergibtanweisung: Variable = Ausdruck:

z.B. VAR = 2**3-4

ii) Vergleichsoperationen:

		Bedeutung:	Beispiel:
"größer als":	.GT.	>	x.GT.y
	.GE.	≥	
	.LT.	<	
	.LE.	≤	
	.NE.	≠	
	.EQ.	=	

iii) BOOLEsche Operationen (logische Operationen):

.NOT.

.AND.

.OR.

2.2.3 Vereinbarungsanweisungen (Spezifikationen)

Vereinbarungsanweisungen dienen zur Vereinbarung der Eigenschaften von Namen im Programm.

i) COMMON-Anweisung: erlaubt, variable Datenelemente in verschiedenen Programmeinheiten gleichzusetzen,

z.B.: `COMMON /ELEM/ A, C, R(10)`

Name des benannten COMMON — Variable

Allein aus der Stellung der Komponente in der Variablenliste ergibt sich deren Wert.

ii) DATA-Anweisung: erlaubt, Variable mit Werten zu versehen,

z.B.: `DATA ALPHA, BETA(2)/5.6., 8.12/`

oder `DATA ALPHA /5.6/, BETA(2) /8.12/`

iii) DIMENSION-Anweisung: erlaubt die Gestalt von Feldern zu vereinbaren,

z.B. : DIMENSION A (6,2,4)

iv) Typanweisung:
Erlaubt den Typ eines Datenelements festzulegen. Datentypen sind:

```
INTEGER
REAL
DOUBLE PRECISION
LOGICAL
CHARACTER
```

2.2.4 Steueranweisungen

i) **Sprunganweisung:** `GO TO.....`

- unbedingte Sprunganweisung: `GO TO n`

z.B.
```
.....................................
 10   A = B + Z
100   B = X + Y
      IF (A - B) 20, 20, 30
 20   Z = A
      GO TO  10
 30   Z = B
      ⋮
.....................................
```

Falls A-B<0: Gehe zu 20

Falls A-B=0: Gehe zu 20

Falls A-B>0: Gehe zu 30

- berechnete Sprunganweisung: `GO TO (i, j, k,..)`

z.B.
```
.....................................
      K = 2
      N = 6
      GO TO (10,110,12,13) N/K
.....................................
```

Hinweis: Da N/K = 3 geht das Programm bei der Ausführung zur Anweisungsmarke 12

Sprunganweisungen (GO TO) sind eher zu vermeiden, da sie der Übersichtlichkeit des Programmcodes abträglich sind. Sprunganweisungen können durch Änderung der Sequenz (Reihenfolge der Anweisungen) oder durch Steueranweisungen für eine Programmverzweigung und Programmschleife ersetzt werden.

ii) **Programmverzweigungen (branching):**

IF-Anweisung zu Fallunterscheidungen

- arithmetische IF-Anweisung: z.B.
```
.....................................
      IF (variab) 10,50,100
.....................................
```

Falls variab <0: Gehe zu 10

" $=0$: " " 50

" >0: " " 100

- logische IF-Anweisung: z.B.
```
IF (I.GE.5) SUM=1.0
```

expression (exp) . true…. oder. false.

- Block-IF-Anweisung: z.B.

```
IF (exp) THEN
      if-Block
ENDIF
```

oder z.B.

```
IF (exp) THEN
      if-Block 1
ELSE
      if-Block 2
END IF
```

iii) **Programmschleifen (loop):**

DO-Anweisung: Eine Gruppe von Anweisungen wird mehrfach ausgeführt,

z.B. `DO 10 i=5,15,3`

Anweisungsnummer	10
erster Wert von i:	5
letzter Wert von i:	15
Inkrement von i:	3

2.2.5 Ein-/Ausgabe von Daten

i) READ-Anweisung: Zum Lesen von Daten: `Read` (logical unit, Anweisungsmarke)

z.B.

```
.....................................
      DIMENSION B(3)
      READ (3,1000)  (B(I), I = 1,3)
1000  FORMAT (3F10.0)
.....................................
```

Im Datensatz steht:

Spalte

```
1.3.....9/........./........./........
        10      2001     -3516
........./........./........./........
```

Falls durch Kommata

getrennt, z.B.: `10.,2001.,-3516.,`

wird das Format ignoriert.

ii) WRITE-Anweisung: Zum Schreiben von Daten: `Write` (logical unit,Anweisungsmarke)

z.B.

```
.....................................
      DIMENSION B(3), D(3)
      WRITE (3,2000) B,D
2000  FORMAT (2F10.2)
.....................................
```

gedruckte Ausgabe im F-Format:

```
Spalte  1.3...7.9/........./
          10.00   2001.00
        3516.00    -30.14
           -.13  45124.56
        ........./........./
```

Beispiel für eine Formatanweisung:

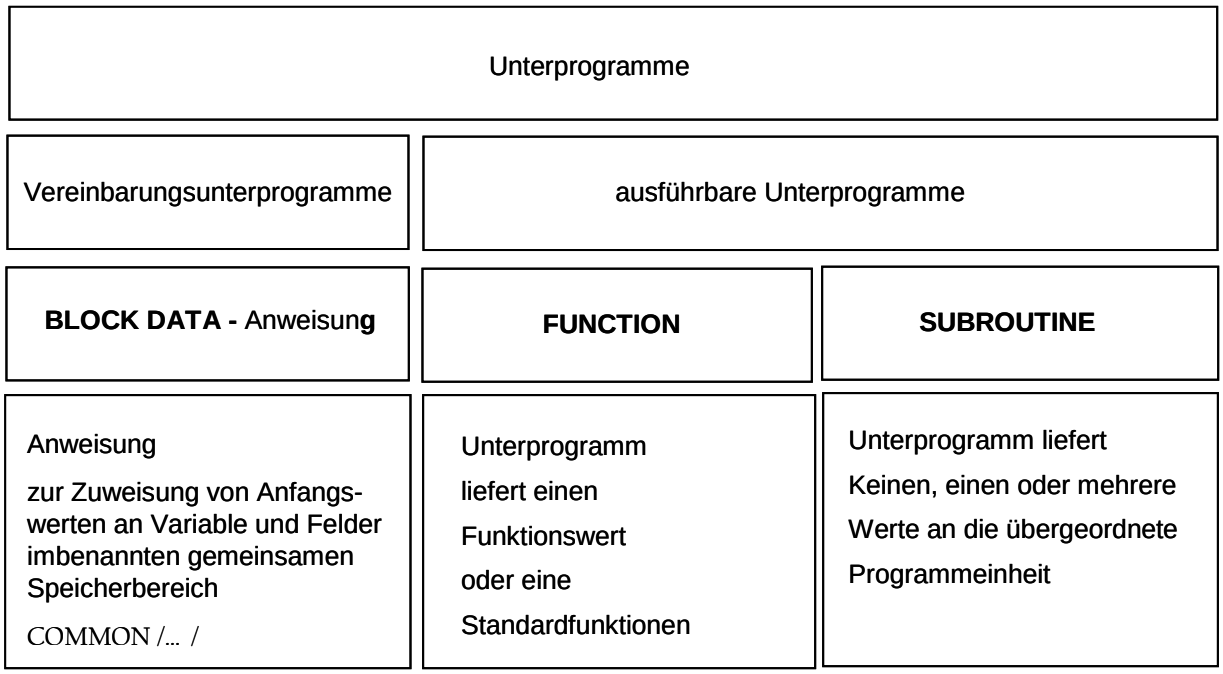
```
.......................................
3000  FORMAT(/,' SUMMARY OF INPUT DATA',/,
     .'BEAM WIDTH ='3F10.3,/,)
.......................................
```

iii) OPEN-Anweisung: Zuordnen einer externen Datei zum FORTRAN-Programm während der Programmausführung.

iv) CLOSE-Anweisung: Freigeben einer zuvor zugeordneten Datei im Lauf der Programmausführung. Auf diese Datei kann vor Ende des laufenden Programms wieder zugegriffen werden, z.B. mit einem Editor oder sie kann zur Einsparung von Speicherplatz gelöscht werden, wenn sie nur temporär während der Programmausführung (z.B. Zwischenspeicherung von Daten) angelegt worden ist.

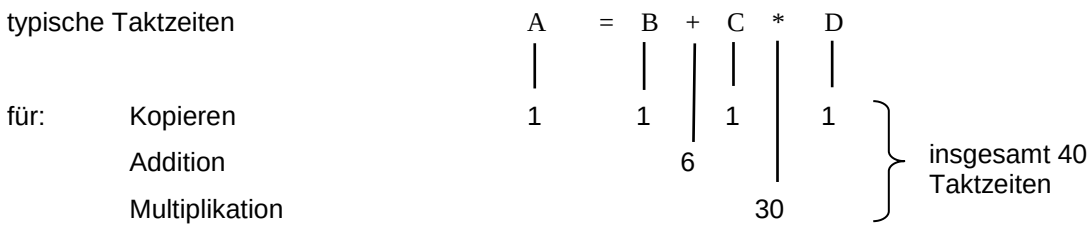
2.2.6 Prozeduren für Funktionen und Subroutinen

Ein Unterprogramm ist eine Programmeinheit, bestehend aus einer Reihe von FORTRAN-Anweisungen. Zwei Arten von Unterprogrammen werden unterschieden:



2.3 Numerische Operationen

Um einen Anhaltspunkt über die Effizienz und Größe eines Algorithmus` zu bekommen, ist es notwendig, die Zahl der numerischen Operationen abschätzen zu können. Die meisten Operationen in FE-Programmen bestehen aus den vier arithmetischen Grundoperationen, wobei Multiplikation und Division ein mehrfaches einer Addition oder Subtraktion ausmachen. Selbst das Kopieren eines Zahlenwertes in die Zentraleinheit kostet Zeit. Als numerische Rechenoperation sollen folgende Arithmetikanweisungen mit Gleitkommazahlen (floating point operations) gelten:



Operationen dieser Gestalt treten in FE-Programmen überwiegend auf, wo die Multiplikation von Matrizen und die Zerlegung der Struktursteifigkeitsmatrix für die Gleichungslösung den Hauptanteil der Rechenzeit ausmachen. Sie werden im englischen Schrifttum als "floating-point operations" (Kurzform:

Flops) bezeichnet, die immer aus einer Addition oder Subtraktion und einer Multiplikation bestehen. Die Leistungsfähigkeit eines Rechners wird häufig als Anzahl der ausgeführten Flops pro Sekunde angegeben (im englischen: **f**loatin point **o**perations **p**er **s**econd) und mit der Kurzform: FLOP/s bzw. mit der üblichen Notation FLOPS bezeichnet.

Typische Leistungseinheiten für Computer sind:

- megaFLOPS = MFLOPS = 10^6 FLOPS (PC's)
- gigaFLOPS = GFLOPS = 10^9 FLOPS (Großrechner)
- teraFLOPS = TFLOPS = 10^{12} FLOPS (Supercomputer)
- pentaFLOPS = PFLOPS = 10^{15} FLOPS (zukünftige Supercomputer)
- exaFLOPS = EFLOPS = 10^{18} FLOPS (in ferner Zukunft)

Für das wissenschaftliche Rechnen darf der Hauptspeicher des Computers nicht zu klein sein, damit die Leistungsfähigkeit (Schnelligkeit) des Prozessors ausgenutzt wird. Ein sehr schneller Rechner mit einem viel zu kleinen Hauptspeicher ist nutzlos, wenn die Gleitkommazahlen nicht vollständig mit Arbeitsspeicher vorgehalten werden können. Als adäquater Hauptspeichergröße st (storage) gilt – siehe [Schönauer 2000] – die Schätzung:

st $\square$ (0,1 bis 1,0) * (# FLOPS)

Ein Rechner mit 1 MFLOPS benötigt 0,1 ÷ 1 MB Hauptspeicher

1 GFLOPS benötigt 0,1 ÷ 1 GB Hauptspeicher

2.4 FORTRAN–Programmierung an Beispielen

Die Programmierung in FORTRAN soll nur auf das Notwendigste in der Veranstaltung beschränkt sein. Exemplarisch sind einige Beispiele ausgewählt worden.

2.4.1 Matrizenoperationen

i) Matrizenmultiplikation und -addition

$\mathbf{C} = \mathbf{A}\,\mathbf{B} + \mathbf{D}$ oder $C_{ij} = \sum_{k=1}^{KK} A_{ik}\, B_{kj} + D_{ij}$

wobei **A** eine $II \;x\; KK$ -Matrix

und **B** eine $KK \;x\; JJ$ -Matrix sein soll.

FORTRAN-Befehle:

```
.....................................
      DO 100 I=1,II
      DO 100 J=1,JJ
      SUM = 0,0
      DO 50 K=1,KK
 50   SUM = SUM + A(I,K) * B(K,J)
100   C(I,J) = SUM + D(I,J)
.....................................
```

Zahl der Rechenoperationen (#op):

DO-Loop 50: # op. = KK

DO-Loop 100: # op. = KK * JJ * II

falls Quadratmatrizen: II = JJ = KK = N:

op. = N^3

ii) Rechenoperationen mit der Transponierten einer Matrix

$\mathbf{C} = \mathbf{A}^T \mathbf{B}$ **A** ist eine 10 x 4-Matrix

B ist eine 10 x 2-Matrix

FORTRAN-Programm: .....

```
      ⋮
      DO 100 I=1,II
      DO 100 J=1,JJ              Schlecht !
100   AT(I,J) = A(J,I)
⋮
.....................................
```

Besser:

```
.....................................
      DO 200 I=1,4
      DO 200 J=1,2
      SUM = 0,0
      DO 300 K=1,10
300   SUM = SUM+A(K,I)*B(K,J)    Indizes vertauschen!
200   C(I,J) = SUM
.....................................
```

iii) Berechnung symmetrischer Matrizen

Symmetrische Matrizen werden dadurch erzeugt, dass die eine Hälfte der Matrix nicht nochmals berechnet wird. Häufiger Fall: Steifigkeitsmatrix eines linearen Systems ist symmetrisch.

Beispiel: $\mathbf{S} = \mathbf{A}^T \mathbf{A}$ erzeugt eine symmetrische Matrix.

```
.....................................
      DO 100 I=1,II
```

```
      DO 100 J=I,II
      SUM = 0,0
      DO 50 K=1,KK
 50   SUM = SUM+A(K,I)*A(K,J)
      S(I,J) = SUM
100   S(J,I) = SUM
.....................................
```

2.4.2 Unterprogramme für FORTRAN

i) Fortran-Programm mit einem FUNCTION-Unterprogramm

Der Ausdruck $x = a + b \cdot c^{3,5}$ taucht wiederholt im Lösungsweg für eine Mechanikaufgabe auf und soll deshalb in einem Unterprogramm während des Programmablaufs berechnet werden, so dass das Unterprogramm wiederholt verwendet (aufgerufen) werden kann. Der Variablenname ist der Name des FUNCTION-Unterprogramms.

Beispiel:

```
.........!.....................! Kommentar
      PROGRAM TEST
      READ 1000, A, B, C     Bedeutet: READ (5,1000) A,B,C
      X = A + FABC(B,C)      5 =  Standardeingabe ≡  Tastatur
      PRINT  2000,  X
1000  FORMAT(3F10.0)
2000  FORMAT(3E15.7)
      END
c.....
c.....                            Hier werden Adressen übergeben!
      FUNCTION FABC(A1,A2)
      FABC = A1*A2**3,5           Kann aus mehreren Zeilen bestehen
      RETURN
      END
.........!...................
```

ii) FORTRAN-Programm mit einem SUBROUTINE-Unterprogramm

Im Gegensatz zum FUNCTION-Unterprogramm können mit einer SUBROUTINE beliebig viele Werte berechnet werden.

Beispiel:

```
.........!...................
      PROGRAM DEMO
      READ 1000 A,B,C          READ von Standardeingabe Tastatur
      CALL SR(A,B,C,X,Y)
      PRINT 2000 X,Y           bedeutet:   PRINT(6,2000)
```

```
c.....                                  6 = Standardausgabe ≡ Bildschirm
      STOP
1000  FORMAT(3F10.0)
2000  FORMAT(2E15.7)
      END
c.....
      SUBROUTINE SR(A1,A2,A3,X1,X2)
      X1=A1*A2**A3                      Bedeutet  x1 = A1 × A2 ^A3
      X2=X1+A1/A3                       Bedeutet  x2 = x1 + A1/A3
      RETURN
      END
......../.................
```

Hinweise:

- Alle Argumente der Subroutine können geändert werden.
- Die Namen der Argumente sind beliebig - nur ihre Stellung innerhalb der Reihenfolge ist wichtig.
- Nur die Speicheradresse - nicht der Wert des Arguments - wird in die Subroutine übergeben.

2.4.3 Ein-/Ausgabe von Daten

Daten werden formatfrei und formatgebunden über die Befehle READ bzw. WRITE eingelesen und ausgegeben. Formatfreie Daten sind Daten in der Form, wie sie in der Rechenanlage intern dargestellt sind und wie sie von der Anlage verarbeitet werden. Diese Form ist oft eine Binärdarstellung und allgemein schwierig zu interpretieren.

Daher arbeitet man bei der Fehlersuche (Debugging) in Programmen mit formatgebundenen Daten.

Die READ- und WRITE-Anweisungen für formatgebundene Ein- und Ausgabe werden in Verbindung mit einer FORMAT-Anweisung verwendet.

```
READ (Gerätenummer, Anweisungsnummer) , Name1, Name2, ...
       Dateinummer

WRITE (        "    ,              "   ), Name1, Name2, ...

                                          Liste von Datennamen,
                                          z.B.: Variablennamen, Datenfelder
                                          Abarbeitung von links nach rechts
Anweisungs-
nummer        FORMAT (Beschreibung 1, Beschreibung 2 ,...)
```

Datenfeldbeschreibungen, bestehend aus Umwandlungsschlüssel (FORMAT-Schlüssel) für

INTEGER-Größen	im I-Schlüssel
REAL-Größen	im E- Schlüssel für Exponentialdarstellung
	im F-Schlüssel für Dezimaldarstellung
	im D-Schlüssel für doppelt genaue Darstellung
sonstige Zeichen	
(alpha-numerisch)	im A-Schlüssel
	im H-Schlüssel für HOLLERITH-Konstante

zum Beispiel:

```
.........................
2000  FORMAT(F10.4)
.........................
```

Hinweise zum F-Schlüssel:

10 = Feldgröße

4 = Dezimalpunktanzeige = Anzahl der Dezimalstellen

Beispiel 1: 3 REAL-Zahlen sollen von der Datei "INPUT" gelesen und danach soll der erste Wert auf die Datei "OUTPUT" ausgedruckt werden.

```
.................................
      PROGRAM INOUT
      OPEN(4,FILE='INPUT')
      OPEN(7,FILE='OUTPUT')
      READ(4,1000) A,B,C
      A = B*C+A
      WRITE(7,2000) A
      CLOSE(4)                        Unter Umständen
      CLOSE(7)                        Unter Umständen
1000..FORMAT(3F10.3)
2000..FORMAT(10X,'ERGEBNIS = ',F10.4)
      STOP
      END
.................................
```

Hinweis zur Formatanweisung, z.B.: 2000 FORMAT(...):

Zuerst 10 Leerzeichen drucken, danach das Wort "Ergebnis" und dann den Zahlenwert als Dezimalzahl.

Das erste Zeichen dient dem Drucker als Steuerzeichen und wird nicht ausgedruckt.

Hinweis zum Formatschlüssel für reelle Zahlen:
Werden bei der Eingabe der Zahlenwerte der Dezimalpunkt im Formatschlüssel angegeben und die Zahlenwerte durch Kommata voneinander getrennt, so wird die Angabe der Feldgröße und die Dezimalpunktanzeige im F-Schlüssel vom Programm ignoriert.

```
                Spalte          1.3...../........./
Eingabedatei:   INPUT           4112.3,20.,0.05

                  Spalte        1.3...../........./........./
Ausgabedatei:   OUTPUT          .........ERGEBNIS =  4113.3000
                                9 Leerzeichen,        F10.4
```

da erstes Zeichen als Steuerzeichen (Vorschub) von manchen Druckern nicht ausgegeben wird.

Beispiel 2: Schreiben von INTEGER- und REAL-Zahlen

```
.........//................................
      K = 12
      A = 3.754
      B = - 4.73
      C = 41.5
      D = - 3.7
      E = 913744.0
      F = 2.423
      WRITE(5,1000) K, A, B, C, D, E, F
1000..FORMAT(I4//F8.2, 2X, F5.2/(2F5.1))
```

↑ neue Zeile

dieser Schlüssel wird wiederholt, falls weniger Datenfeldbeschreibungen als auszugebende Größen definiert sind.

Ausdruck:	Spalte	Daten-	FORMAT
	1.3.5..../...../...../..	element	
1. Zeile	12	K	I4
2. Zeile			
3. Zeile	3.75 -4.73	A,B	F8.2,2X,F5.2
4. Zeile	41.5 -3.7	C,D	2F5.1
5. Zeile	***** 2.4	E,F	2F5.1
	...../...../...../..		

Dezimalstellen vom Datenelement "F" werden abgeschnitten.

Hinweise: INTEGER- und REAL-Größen sind rechtsbündig.

Beispiel 3: Einlesen einer Matrix mit Hilfe eines Unterprogramms

Die Matrix hat die Größe NR x NC

Maximal 6 Matrixelemente werden pro Zeile eingelesen.

Es werden zeilenweise immer die ersten 6 Werte mit einer impliziten DO-Schleife eingelesen.

```
.....................................
      SUBROUTINE MREAD(A,NR,NC)
      DIMENSION A(NR,NC)
c.....READ MATRIX FROM INPUT FILE
      DO 100 J=1,NC,6
      JH = J + 5
      DO 100 I=1,NR
      READ(6,1000) (A(I,K), K=J,JH)
100   CONTINUE
c.....
      RETURN
1000  FORMAT(6F10.0)
      END
.....................................
```

Beispiel 4: Ausgabe einer Matrix mit Hilfe eines Unterprogramms

```
.........................................
      SUBROUTINE MPRINT (A,NR,NC,NAME)
      CHARACTER*6 NAME
      DIMENSION  A(NR,NC)
c     GENERAL MATRIX PRINT SUBROUTINE ......
      WRITE (3,2000) NAME
c.....
      DO 100 J=1,NC,5
      JH = J + 4
      IF (JH.GT.NC) JH = NC
      WRITE(3,2001) (N,N = J,JH)
      Do 100 I=1,NR
100   WRITE(3,2002) I,(A(I,K),K = J,JH)
      RETURN
C.....
2000  FORMAT(/,'MATRIX ',1A2)
2001  FORMAT(/8X,5I14)
2002  FORMAT(I4,4X,5E14.7)
      END
.........................................
```

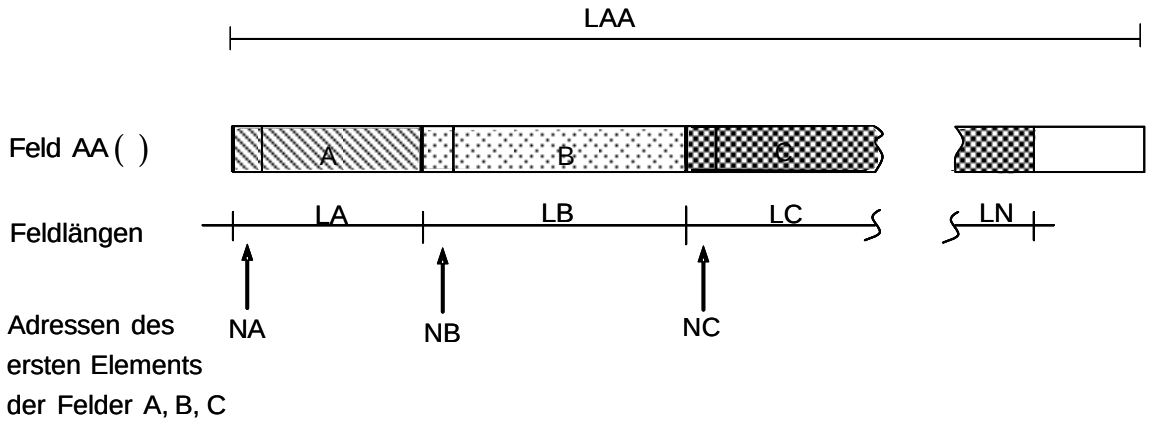
2.5 Dynamische Speicherplatzverwaltung

Im Folgenden soll das Problem gelöst werden, dass Felder bei der Ausführung verschiedener Berechnungsaufgaben unterschiedliche Größen annehmen. Die Struktursteifigkeitsmatrix **K**, die sich aus den Elementsteifigkeitsmatrizen **k** zusammensetzt, ist z.B. von der Anzahl der Freiheitsgrade abhängig, mit denen die Verformungen eines Bauteils beschrieben werden.

Es gibt drei Möglichkeiten, Felder nach Übersetzen und Binden eines FORTRAN-Programms im Nachhinein während der Ausführung variabel anzulegen.

i) Halbdynamische Felder

Bevor ein Feld in FORTRAN benutzt wird, muss es fest dimensioniert werden, d.h. der Länge des Feldes muss ein fester Zahlenwert vorgegeben werden, damit das Feld u.U. im unbenannten COMMON vorgehalten werden kann. Die Zahl der Knoten, Elemente, Lastfälle usw. ist jedoch eine problemabhängige Größe und bestimmt die Länge der Felder für die Speicherung der Knotenkoordinaten, Elementeigenschaften, Lastvektoren usw., denn man will die Felder nicht unnötig lang machen und kleine und große Strukturen mit demselben Programm berechnen. Daher werden in jedem Rechenlauf die Feldgrößen zuerst berechnet, wozu die notwendigen Daten wie Zahl der Knoten, Freiheitsgrade pro Knoten, Zahl der Elemente etc. zuvor eingelesen werden müssen. Danach werden die Felder entsprechend den Größenanforderungen des FE-Problems dimensioniert, wobei solche von Rechenlauf zu Rechenlauf veränderlichen Feldlängen nur bei der Felddimensionierung in Unterprogrammen vorkommen dürfen. Am Beispiel der Ein- und Ausgabe zweier Matrizen der Größe $N \times N$ und $N \times L$ soll das Vorgehen demonstriert werden.



Beispiel : Ein- und Ausgabe der Matrizen A und B mit einem FORTRAN-Programm und dynamischer Speicherplatzverwaltung

```
.............................................
      PROGRAM IOCHK
      DIMENSION AA(100)
c.....PROGRAM TO ILLUSTRATE FILE INPUT OPTIONS
      OPEN(6,'INPUT')
      OPEN(7,'OUTPUT')
c.....READ MATRIX SIZES
      READ(6,1000) NN,LL
      WRITE(7,2000) NN,LL
c.....DYNAMIC STORAGE ALLOCATION
      NA = 1
      NB = NA + NN * NN
      NT = NB + NN * LL
      IF (NT.GT.100) WRITE(3,3000)
c.....READ AND WRITE MATRICES WITH SUBROUTINES
      CALL MREAD(AA(NA),NN,NN)
      CALL MPRINT(AA(NA),NN,NN,'A')
      CALL MREAD(AA(NB),NN,LL)
      CALL MPRINT(AA(NB),NN,LL,'B')
      STOP
c.....
1000  FORMAT(2I5)
2000  FORMAT(/,'DRUCKEN DER MATRIX ',
     . 'MIT ',I5,' ZEILEN'
     . 'UND ',I5,' SPALTEN')
3000  FORMAT (/,'NICHT GENUEGEND SPEICHER')
      END
.............................................
```

Im obigen Beispiel wird gezeigt, wie eine halbdynamische Speicherplatzverwaltung unter der Programmiersprache FORTRAN77 realisiert werden kann. Dazu wird also im Hauptprogramm ein so genannter „BLANK COMMON“ oder ein eindimensionales Feld - hier `AA(100)` - mit fester Feldgrenze statisch angelegt. Innerhalb seiner Grenzen kann dann ein Unterprogramm Felder mit variabler Länge dynamisch verwalten. Dieses Vorgehen der Verwaltung variabler Felder wird vor allem in älteren FORTRAN-Programmen verwendet.

ii) Alternativlösung: „Platzhalter“ in FORTRAN77

In FORTRAN77 kann anstelle der halbdynamischen Speicherplatzverwaltung eine alternative Lösung erfolgen. Hierzu ist ein so genannter „Platzhalter“ mit entsprechender Dimension und festen Feldgrenzen im Programm anzulegen. In den danach aufgerufenen Unterprogrammen müssen die durch Platzhalter angelegten Felder der Übergabeliste des Unterprogramms beigefügt werden. In den Unterprogrammen selbst können diese Felder dann variabel mit den zuvor bestimmten Feldgrenzen angelegt werden – siehe Programm „MATRIXADD“.

Beispiel: Einlesen und Addieren zweier Matrizen

```
      ..............................................................
      PROGRAM MATRIXADDITION
C*************************************************************
C     Programm zur Demonstration dynamisch angelegter Felder
C     am Beispiel der Matrixaddition
C*************************************************************

C     Variablenvereinbarung
C     ***** Variable *****
      implicit none
      integer mm
C
C**** Platzhalter mit beliebig großer, aber fester Feldgrösse
C
      real dmata(100,100), dmatb(100,100),dmatc(100,100)
C*************************************************
C     *** prozeduraler Teil ***
C     *** Matrixgroesse m eingeben ***
C
      write(*,999)
 999  format('Gib Matrixgroesse ein')
      read (*,1000) mm
 1000 format (i5)
      write(*,1001) mm
 1001 format (5X, 'Matrixgroesse m = ', i5)
      if (mm .gt. 0) then
         call addmat(mm,dmata,dmatb,dmatc)
```

```
      else
         write(*,1002)
 1002     format('leere Matrix, keine Rechnung moegl.')
      endif
      write (*, 1100)
 1100 format (' Programm fertig')
      end
C**** Ende Hauptprogramm *****************
C
c
c**** Unterprogramm
c
      subroutine addmat(m,mata,matb,matc)
      integer m,i,j
C
C***  Neudimensionierung der Matrizen
C
      real mata(m,m),matb(m,m),matc(m,m)

c
C**** Einlesen der Matrizen mata und matb aus Eingabedatei 'INPUT'
      open(6,file='INPUT')
      call MREAD(mata,m,m)
      call MREAD(matb,m,m)
      close(6)
c
C**** Addition der Matrizen
      do 100 i=1,m
         do 200 j=1,m
            matc(i,j)=mata(i,j)+matb(i,j)
 200     continue
 100  continue
c
C**** Ausgabe der Ergebnisse auf die Ausgabedatei `OUTPUT`
      open(7,file='OUTPUT')
      call MPRINT(mata,m,m,'A     ')
      call MPRINT(matb,m,m,'B     ')
      call MPRINT(matc,m,m,'C=A+B ')
      close(7)
```

```
      return
      end subroutine
.....................................................................
```

iii) Volldynamische Felder in FORTRAN90

Im Abschnitt 2.5 i) wurde gezeigt, wie eine halbdynamische Speicherplatzverwaltung unter FORTRAN77 zum Anlegen von Feldern mit problemabhängiger Länge eingesetzt werden kann. Mit der Einführung der Version FORTRAN90 ist es möglich geworden, Felder erst während der Laufzeit volldynamisch zu handhaben, ohne dass mit statischen „Platzhaltern“ für variable Felder oder statischem „blank COMMON BLOCK“ und halbdynamischen Feldern gearbeitet werden muss. Faktisch bedeutet diese Option in FORTRAN90, dass erst zur Laufzeit eines Programms die Größe eines ARRAYs (Feldes) problemabhängig gewählt werden kann. Das Feld ist bei der Variablendeklaration als „ALLOCATABLE“ zu vereinbaren:

Beispiel:

```
.......ALLOCATABLE A(:,:)
```

Hier wird nur die Dimension (im Beispiel: n=2) des Feldes A festgelegt. Die noch unbekannten Dimensionsgrößen werden durch die Doppelpunkte vertreten.

Wird das Feld A während der Programmausführung mit der Größe $m \times n$ benötigt, so schreibt man:

Beispiel:

```
.......ALLOCATE (A(m,n))
```

Dem Feld A wird durch die ALLOCATE-Anweisung der benötigte Speicherplatz durch die zuvor bestimmten Grenzen m und n zugewiesen.

Wird das Feld im Weiteren nicht mehr gebraucht, so kann der reservierte Speicher für das Feld durch die DEALLOCATE-Anweisung freigegeben werden. Dasselbe geschieht ebenfalls durch die Befehle END oder RETURN.

Beispiel: Einlesen und Addieren zweier Matrizen:

```
      .....................................................................
      PROGRAM MATRIXADDITION
!*************************************************************
!     Programm zur Demonstration dynamisch angelegter Felder
```

```
!     in-- FORTRAN 90 --am Beispiel der Matrixaddition
!*********************************************************

!     Variablenvereinbarung
!     ***** Variable *****
      implicit none
      integer mm

!**************************************
!     *** prozeduraler Teil ***
!     *** Matrixgroesse m eingeben ***
      write(*,999)
 999  format('Gib Matrixgroesse ein')
      read (*,1000) mm
 1000 format (i5)
      write(*,1001) mm
 1001 format (5X, 'Matrixgroesse m = ', i5)
      if (mm .gt. 0) then
         call addmat(mm)
      else
         write(*,1002)
 1002     format('leere Matrix, keine Rechnung moegl.')
      endif
      write (*, 1100)
 1100 format (' Programm fertig')
      end
!**** Ende Hauptprogramm ****************
!
!**** Unterprogramm
!
      subroutine addmat(m)
!
      integer m,i,j
       character*6  nameA, nameB, nameC
!******** Deklaration dynamischer Arrays*************
      real, allocatable mata(:,:),matb(:,:),matc(:,:)
!************************************************
!
!**** Einlesen der Matrizen mata und matb
```

```
!     **** Zeilenweises einlesen von Matrix mata
      write(*,99)
 99   format ('Werte-Eingabe Matrix a')
      allocate (mata(m,m))                    ! Speicher zuordnen
      do 200 i = 1,m,1
      do 100 j = 1,m,1
      read(*,*) mata(i,j)
100   continue
200   continue
!
!     **** Zeilenweises Einlesen von Matrix matb
      write(*,299)
299   format('Werte-Eingabe von Matrix b')
      allocate (matb(m,m))                    ! Speicher zuordnen
      do 400 i = 1,m,1
      do 300 j = 1,m,1
      read(*,*) matb(i,j)
300   continue
400   continue
!
!**** Addition der Matrizen
      allocate (matc(m,m))                    ! Speicher zuordnen
      do 600 i=1,m
         do 500 j=1,m
            matc(i,j)=mata(i,j)+matb(i,j)
 500     continue
 600  continue
!
!**** Ausgabe der Ergebnisse
!     **** Zeilenweises Ausgeben von Matrix matc
      Write (*,*) 'Werte von matc, zeilenweise'
      do 800 i = 1,m,1
      write (*,3000) i
 3000 format (5X, 'Die Elemente der Zeile ',I5)
      do 700 j = 1,m,1
      write (*, 3001) matc(i,j)
 3001 format(5X,f6.2)
 700  continue
 800  continue
```

```
      deallocate (mata,matb,matc)           ! Speicherplatz freigeben
      return
      end subroutine
.....................................................................
```

2.6 Programmierregeln

1. Felder, deren Werte sich bei der Berechnung ändern, sollen immer über die Argumentenliste der SUBROUTINEN übergeben werden.

2. Keine READ/WRITE-Befehle in SUBROUTINEN, wo gerechnet wird, sondern in eigens dafür vorgesehen Ein-/Ausgaberoutinen.

3. Variablen, die verändert werden, aus Gründen der Überschaubarkeit möglichst über Argumentenliste der SUBROUTINE übergeben und nicht über COMMON-Blöcke, außer es handelt sich um "lokale" COMMON-Blöcke.

4. Umfangreichere Berechnungsschritte in einzelnen SUBROUTINEN ausführen, die von einem gemeinsamen Steuerprogramm gerufen werden.

5. Keine "Spaghetti"-Unterprogramme schreiben.

6. Sprung-(GOTO)-Anweisungen nach Möglichkeit auf ein Minimum reduzieren. Dafür die Befehle IF...THEN...ELSEIF...ENDIF verwenden

Am Beispiel der Lösung regulärer linearer Gleichungssysteme sollen mathematische Algorithmen aufgestellt, FORTRAN-Programme geschrieben und die Zahl der numerischen Operationen ermittelt werden.

3 Lösung linearer Gleichungssysteme

Die Verfahren zur Lösung linearer Gleichungssysteme

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

lassen sich in direkte Verfahren (explizite Lösung) und iterative Methoden (Näherungsverfahren) einteilen. Für sehr große Gleichungssysteme haben sich mittlerweile die iterativen Methoden durchgesetzt, falls die Koeffizientenmatrix bezüglich der Inversion nicht schlecht konditioniert ist. Im Rahmen der Einführungsvorlesung „Computational Mechanics“ soll ein direktes Lösungsverfahren vorgestellt werden. Unter die direkten Lösungsverfahren fallen:

1) CRAMERsche Regel (~ 1750):

   Gut für die Handrechnung sehr kleiner Gleichungssysteme, da die CRAMERsche Regel sehr rechenintensiv ist, denn:

   Zahl der Rechenoperationen: # op. = N!

2) GAUSSsche Elimination (~ 1826):

   Geeignet für Rechner, falls das Gleichungssystem nur für eine "rechte Seite" **b** gelöst werden muss.

   Zahl der Rechenoperationen: # op. ~ N^3

3) Faktorisierung der Koeffizientenmatrix (CHOLESKY ~ 1916):

   Geeignet für Rechner, wenn das Gleichungssystem für verschiedene "rechte Seiten" **b** zu lösen ist.

   Zahl der Rechenoperationen: # op. ~ N^3,

wobei N die Zahl der Gleichungen ist.

In Anbetracht der riesigen Anzahl von Rechenoperationen (N!) für die CRAMERsche Regel - selbst bei relativ kleinen Gleichungssystemen - kommt nur das GAUSSsche Eliminationsverfahren oder die Faktorisierung nach CHOLESKY in der FEM in Frage.

Aus der linearen Algebra ist bekannt, dass die Lösung eines Gleichungssystems

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \tag{2.5.1}$$

mit **A** als $N \times N$ -Matrix existiert, falls eine der Aussagen im folgenden Theorem gilt.

Satz: Folgende Aussagen sind gleichwertig:

i) $\mathbf{A}^{-1}$ existiert, wobei $\mathbf{A}^{-1}$ die Inverse zu $\mathbf{A}$ ist.

ii) Die Determinante der Matrix $\mathbf{A}$ ist von Null verschieden: $\det \mathbf{A} \neq 0$.

iii) Es gibt keinen vom Nullvektor $\mathbf{0}$ verschiedenen Vektor $\mathbf{x}$ mit der Eigenschaft $\mathbf{A}\,\mathbf{x} = \mathbf{0}$.

iv) Die Zeilen von $\mathbf{A}$ sind linear unabhängig.

v) Die Spalten von $\mathbf{A}$ sind linear unabhängig.

Def.: Falls obige Bedingungen gelten, heißt die Matrix $\mathbf{A}$ **nicht singulär** oder **invertierbar**.

Def.: Eine symmetrische $N\,x\,N$-Matrix $\mathbf{A}$ - d.h. $\mathbf{A} = \mathbf{A}^{T}$ - heißt **positiv definit**, falls gilt:

$$\mathbf{x}^{T}\mathbf{A}\,\mathbf{x} > 0$$

für alle Vektoren $\mathbf{x} \neq \mathbf{0}$.

Satz: Falls $\mathbf{A}$ positiv definit ist, ist $\mathbf{A}$ nicht singulär.

Def.: Unter den Pivotelementen versteht man die Divisoren (Teiler), d.h. die Nenner in den Multipolikatoren, die während der GAUSSschen Eliminationsmethode auftreten.

Die Pivotelemente bilden die Hauptdiagonalelemente in der oberen Dreiecksmatrix $\mathbf{U}$ der faktorisierten Koeffizientenmatrix $\mathbf{A} = \mathbf{LU}$ - s. Kap. 2.5.4.

Satz: Falls $\mathbf{A}$ positiv definit ist, dann gelten die folgenden Aussagen:

i) Alle "Pivotelemente" in der Elimination sind positiv definit.

ii) Alle führenden Hauptabschnittsuntermatrizen sind positiv definit.

iii) Die Determinante von $\mathbf{A}$ $(\det \mathbf{A} > 0)$ und die aller führenden Hauptabschnittsuntermatrizen sind positiv.

Die Erklärung und der Beweis von Aussage i) folgen später, da beides für das CHOLESKY-Verfahren im nächsten Kapitel von großer Bedeutung ist.

3.1 Gleichungssysteme mit dreieckförmig besetzter Koeffizientenmatrix

Ein lineares Gleichungssystem mit dreieckförmig besetzter Koeffizientenmatrix ist besonders einfach zu lösen.

$$\mathbf{G}\,\mathbf{x} = \mathbf{b} \tag{2.5.2}$$

mit

$$\mathbf{G} = \begin{bmatrix} g_{11} & 0 & 0 & \dots & 0 \\ g_{21} & g_{22} & 0 & \dots & 0 \\ g_{31} & g_{32} & g_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & g_{n3} & \dots & g_{nn} \end{bmatrix}$$

Satz: **G** ist nicht singulär, falls alle Hauptdiagonalelemente $g_{ii} \neq 0,\ i = 1,2,\dots,n$ von Null verschieden sind.

Beweis: $\det \mathbf{G} = g_{11}\, g_{22}\, g_{33} \dots g_{nn} \neq 0$ falls alle $g_{ii} \neq 0$ *für* $i = 1,2,\dots,n$.

Gleichungssystem:

$$g_{11}\, x_1 = b_1 \tag{2.5.3}$$

$$g_{21}\, x_1 + g_{22}\, x_2 = b_2 \tag{2.5.4}$$

$$g_{31}\, x_1 + g_{32}\, x_2 + g_{33}\, x_3 = b_3 \tag{2.5.5}$$

$$\vdots \qquad\qquad \vdots$$

$$g_{n1}\, x_1 + g_{n2}\, x_2 + g_{n3}\, x_3 + \dots + g_{nn}\, x_n = b_n \tag{2.5.6}$$

Aus Gl. (2.5.3) folgt: $x_1 = \dfrac{b_1}{g_{11}}$

Einsetzen in Gl. (2.5.4): $x_2 = \left(b_2 - g_{21}\, x_1\right) / g_{22}$

$\vdots \qquad\qquad \vdots$

Für die $i-te$-Gleichung gilt:

$$x_i = \frac{\left(b_i - g_{i1}\, x_1 - g_{i2}\, x_2 - \ldots - g_{i,i-1}\, x_{i-1}\right)}{g_{ii}}$$

$$\boxed{x_i = \frac{1}{g_{ii}} \left(b_i - \sum_{j=1}^{i-1} g_{ij}\, x_j \right)} \qquad (2.5.7)$$

Das FORTRAN-Programm für die Lösung des Gleichungssystems kann wie folgt aussehen:

```
...../.........................................
......SUBROUTINE SOLTRI(G,B,NEQ)
......DIM B(NEQ), G(NEQ,NEQ)
......DO 100  I=1,NEQ
......  IF (G(I,I).EQ.0) RETURN
......  I1 = I-1
......  DO 200  J=1,I1
200       B(I) = B(I) - G(I,J)*B(J)
100     B(I) = B(I) / G(I,I)
......RETURN
...../.........................................
```

Beachte:

- Der Lösungsvektor **x** wird in der Subroutine SOLTRI auf dem Feld für den Vektor der rechten Seite **b** gespeichert.
- Um Gleichungssysteme beliebiger Ordnung lösen zu können, werden die Anfangsadressen der Felder B und G durch die Argumentenliste der FORTRAN-Subroutine übergeben, wo variable Dimensionierung im Gegensatz zu einem benannten COMMON-Block möglich ist.

Die Zahl der Rechenoperationen zur Lösung des Gleichungssystems (2.5.2) bis (2.5.7), abgesehen von den NEQ-Divisionen, wird errechnet zu:

Gl. (2.5.3):	0
Gl. (2.5.4):	1
Gl. (2.5.5):	2
Gl. (2.5.6):	3
⋮	⋮
Gl. (2.5.7):	n-1

(2.5.8)

Summe: $$\#\,op. = \sum_{i=1}^{n}(i-1) = \frac{n(n-1)}{2} \cong \frac{1}{2}n^2$$

Die Zahl der Rechenoperationen zur Lösung eines Gleichungssystems mit dreieckförmig besetzter Koeffizientenmatrix wächst mit dem Quadrat der Anzahl der Gleichungen N.

Ähnlich einfach lassen sich Gleichungssysteme mit einer oberen Dreiecksmatrix als Koeffizientenmatrix lösen. Zur Elimination der Unbekannten wird dann mit der letzten Gleichung begonnen.

3.2 CHOLESKY – Zerlegung für GLSe mit positiv definiter Koeffizientenmatrix

3.2.1 Grundlagen der CHOLESKY-Zerlegung

Mit der CHOLESKY-Zerlegung (CHOLESKY war französischer Vermessungsingenieur) lässt sich eine positiv definite Matrix **A** in das Produkt einer unteren und oberen Dreiecksmatrix zerlegen.

Die Steifigkeitsmatrix **K** eines brauchbaren (keine verformungslosen Kinematiken!) Systems auf der Grundlage der linearen Elastizitätstheorie ist immer positiv definit. Daher ist jedes brauchbare, linear elastische System lösbar.

Wichtige Sätze als Grundlage für das CHOLESKY-Verfahren

Satz: Wenn **A** positiv definit ist, dann ist **A** nicht singulär (invertierbar).

Satz: Wenn **A** positiv definit ist, dann hat das lineare Gleichungssystem $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ genau eine Lösung.

Satz: **M** sei eine reelle, nicht singuläre $N\,x\,N$-Matrix , und es gilt $\mathbf{A} = \mathbf{M}\mathbf{M}^T$. Dann ist **A** positiv definit.

Beispiel: Die Matrix

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

ist nicht singulär, denn $\det \mathbf{M} = 1 \cdot 1 \cdot 1 = 1 \neq 0$

$$\mathbf{A} = \mathbf{M}\mathbf{M}^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \quad \text{ist positiv definit}$$

Satz: CHOLESKY-Zerlegung

Jede positiv definite Matrix **A** kann auf genau eine Art in ein Produkt

$$\mathbf{A} = \mathbf{G}\,\mathbf{G}^{\mathrm{T}} \tag{2.5.9}$$

zerlegt werden, so dass **G** eine untere Dreiecksmatrix mit positiven Hauptdiagonalgliedern ist. Man nennt **G** den CHOLESKY-Faktor von **A**.

Im obigen Beispiel ist die Matrix **M** der CHOLESKY-Faktor von **A**.

3.2.2 CHOLESKY – Algorithmus

Zum Auffinden eines praktischen Algorithmus` für die CHOLESKY-Zerlegung der Matrix **A** wird das Produkt $\mathbf{A} = \mathbf{G}\mathbf{G}^{\mathrm{T}}$ betrachtet.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} g_{11} & 0 & 0 & \dots & 0 \\ g_{21} & g_{22} & 0 & \dots & 0 \\ g_{31} & g_{32} & g_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & g_{n3} & \dots & g_{nn} \end{bmatrix} \cdot \begin{bmatrix} g_{11} & g_{21} & g_{31} & \dots & g_{n1} \\ 0 & g_{22} & g_{32} & \dots & g_{n2} \\ 0 & 0 & g_{33} & \dots & g_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & g_{nn} \end{bmatrix} \tag{2.5.10}$$

Das Matrixelement a_{ij} ist das (innere) Produkt der i-ten Reihe von **G** mit der j-ten Spalte von $\mathbf{G}^{\mathrm{T}}$. Die erste Spalte von $\mathbf{G}^{\mathrm{T}}$ hat nur ein von Null verschiedenes Element.

$$a_{i1} = g_{i1} \cdot g_{11} + g_{i2} \cdot 0 + g_{i3} \cdot 0 + \ldots = g_{i1}\, g_{11} \tag{2.5.11a}$$

Im Fall $i = 1$ gilt: $a_{11} = g_{11}^2$

Von den beiden Lösungen dieser Gleichung für g_{11} ist die positive Quadratwurzel zu nehmen.

$$g_{11} = +\sqrt{a_{11}} \quad , \tag{2.5.11b}$$

denn alle Hauptdiagonalglieder g_{ii} von **G** müssen positiv sein, wenn **A** positiv definit ist. Mit Hilfe von g_{11} folgt aus Gl. (2.5.11a):

$$g_{i1} = \frac{a_{i1}}{g_{11}} \quad \text{für} \quad i = 2, 3, ..., n \tag{2.5.11c}$$

Damit sind alle Elemente der ersten Spalte von **G** und auch der ersten Zeile von $\mathbf{G}^T$ bestimmt. Die Elemente a_{i2} der zweiten Spalte von **A** entstehen aus dem Produkt

$$a_{i2} = g_{i1}\, g_{21} + g_{i2}\, g_{22}\ , \tag{2.5.12a}$$

denn bis auf die ersten beiden Elemente in der zweiten Spalte von $\mathbf{G}^T$ sind alle anderen Elemente gleich null.

Im Fall $i = 2$ gilt: $a_{22} = g_{21}\, g_{21} + g_{22}\, g_{22}$

Als Lösung für g_{22} kommt nur die positive Quadratwurzel in Betracht.

$$\Leftrightarrow \quad g_{22} = \sqrt{a_{22} - g_{21}^2}\ , \tag{2.5.12b}$$

denn g_{21} ist aus Gl. (2.5.11c) bereits bekannt. Ebenso bestimmt sind die Elemente g_{i1} aus Gl. (2.5.11c), so dass Gl. (2.5.12a) nach dem unbekannten Matrixelement g_{i2} aufgelöst werden kann.

$$g_{i2} = \frac{1}{g_{22}} \left(a_{i2} - g_{i1}\, g_{21}\right) \text{ für} \quad i = 3, 4, 5, ..., n \tag{2.5.12c}$$

Da $g_{12} = 0$ ist, sind alle Elemente der zweiten Spalte von **G** und der zweiten Zeile von $\mathbf{G}^T$ bekannt. Entsprechend können die Elemente der 3. Spalte von **G** bestimmt werden, indem die i-te Zeile von **G** mit der 3. Spalte von $\mathbf{G}^T$ multipliziert wird.

$$a_{i3} = g_{i1}\, g_{31} + g_{i2}\, g_{32} + g_{i3}\, g_{33} \tag{2.5.13a}$$

Für $i = 3$ und mit den bekannten Elementen g_{31} aus Gl. (2.5.11c) und g_{32} aus Gl. (2.5.12c) folgt aus Gl. (2.5.13a):

$$a_{33} = g_{31}^2 + g_{32}^2 + g_{33}^2$$

das unbekannte Element g_{33}, das positiv sein muss.

$$g_{33} = \sqrt{a_{33} - g_{31}^2 - g_{32}^2} \tag{2.5.13b}$$

Damit lassen sich alle übrigen Elemente der 3. Spalte von **G** aus der Vorschrift (2.5.13a) nach deren Umstellung berechnen.

$$g_{i3} = \frac{1}{g_{33}} \left(a_{i3} - g_{i1}\, g_{31} - g_{i2}\, g_{32}\right) \tag{2.5.13c}$$

Die Struktur der Gleichung (2.5.11a) bis (2.5.13c) legt es nahe, dass eine Rekursionsformel für die Bestimmung des Matrixelements g_{ij} in Zeile i und Spalte j der Matrix **G** gefunden werden kann. Unter der Annahme, dass alle Elemente in den ersten $j-1$ -Spalten bekannt sind, errechnet sich das Element a_{ij} der Matrix **A** aus dem inneren Produkt der Zeile i von Matrix **G** und Spalte j von Matrix $\mathbf{G}^T$

$$a_{ij} = g_{i1}\, g_{j1} + g_{i2}\, g_{j2} + \quad \ldots \quad + g_{i,j-1}\, g_{j,j-1} + g_{ij}\, g_{jj}\,. \tag{2.5.14a}$$

denn nur die ersten j-Terme der Spalte j in $\mathbf{G}^T$ sind ungleich null. Alle Elemente der Gl. (2.5.14a) stammen aus den ersten j-Spalten von **G**, wobei die Elemente der ersten (**j** - 1) -Spalten bekannt sind. Somit sind in Gl. (2.5.14a) nur g_{ij} und g_{jj} unbekannt. Im Fall $i = j$ folgt aus Gl. (2.5.14a) die Bestimmungsgleichung für g_{ii}.

$$a_{jj} = g_{j1}^2 + g_{j2}^2 + \ldots + g_{j,j-1}^2 + g_{jj}^2 = \sum_{k=1}^{j-1} g_{jk}^2 + g_{jj}^2$$

Sie kann aufgelöst werden nach:

$$g_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} g_{jk}^2} \tag{2.5.14b}$$

Anhand von Gl. (2.5.14b) werden aus Gl. (2.5.14a) alle restlichen Elemente g_{ij} der Spalte j von **G** aus der Beziehung

$$\boxed{g_{ij} = \frac{1}{g_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} g_{ik}\, g_{jk} \right)} \qquad \text{für } i = j+1, j+2, \quad \ldots, \quad n \tag{2.5.14c}$$

berechnet. Man bezeichnet diesen Algorithmus als das CHOLESKY-Verfahren oder die **Quadratwurzel-Methode.**

Hinweise zum CHOLESKY-Verfahren:

- Das CHOLESKY-Verfahren kann nur auf positiv definite Matrizen angewendet werden, da nur in diesem Fall der Radikant in Gl. (2.5.14b) immer positiv ist.

- Andererseits ist das CHOLESKY-Verfahren die beste bekannte numerische Methode, um festzustellen, ob eine Matrix positiv definit ist. Wenn die untersuchte Matrix nicht positiv definit ist, versagt das CHOLESKY-Verfahren immer aufgrund der Tatsache, dass der Radikant in Gl. (2.5.14b) nicht positiv ist.

- Das Element a_{ij} der Matrix **A** wird nur einmal benutzt, um g_{ij} zu berechnen. Daher kann a_{ij} mit g_{ij} überschrieben werden.

- Die CHOLESKY-Zerlegung bewahrt die Bandstruktur einer Matrix, welche durch Inversion der Koeffizientenmatix verloren gehen würde.

- Die vorgestellte Version der CHOLESKY-Zerlegung ist die innere Produktformulierung. Daneben existieren noch zwei weitere Versionen für die Zerlegung, von denen im Hinblick auf Vektorrechner die so genannte äußere Produktformulierung deutliche Vorteile gegenüber der inneren Produktversion hat.

Beispiel: CHOLESKY-Zerlegung der Matrix

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 4 & 2 \\ -2 & 10 & -2 & -7 \\ 4 & -2 & 8 & 4 \\ 2 & -7 & 4 & 7 \end{bmatrix} = \begin{bmatrix} g_{11} & & & \\ g_{21} & g_{22} & & \\ g_{31} & g_{32} & g_{33} & \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \cdot \begin{bmatrix} g_{11} & g_{21} & g_{31} & g_{41} \\ & g_{22} & g_{32} & g_{42} \\ & & g_{33} & g_{43} \\ & & & g_{44} \end{bmatrix}$$

$$g_{11} = \sqrt{a_{11}} = 2$$

$$g_{21} = a_{21}/g_{11} = -1$$

$$g_{31} = a_{31}/g_{11} = 2$$

$$g_{41} = a_{41}/g_{11} = 1$$

$$g_{22} = \sqrt{a_{22} - g_{21}^2} = \sqrt{10 - 1^2} = 3$$

$$g_{32} = \frac{a_{32} - g_{31}\, g_{21}}{g_{22}} = \frac{-2 - (2)(-1)}{3} = 0$$

$$g_{42} = \frac{a_{42} - g_{41}\, g_{21}}{g_{22}} = \frac{-7 - 1(-1)}{3} = -2$$

$$g_{33} = \sqrt{a_{33} - g_{31}^2 - g_{32}^2} = \sqrt{8 - 2^2 - 0^2} = 2$$

$$g_{43} = \frac{a_{43} - g_{41}\, g_{31} - g_{42}\, g_{32}}{g_{33}} = \frac{4 - 1 \cdot 2 - (-2)\, 0}{2} = 1$$

$$g_{44} = \sqrt{a_{44} - g_{41}^2 - g_{42}^2 - g_{43}^2} = \sqrt{7 - 1^2 - (-2)^2 - 1^2} = 1$$

$$\mathbf{G} = \begin{bmatrix} 2 & & & \\ -1 & 3 & & \\ 2 & 0 & 2 & \\ 1 & -2 & 1 & 1 \end{bmatrix}$$

Da der CHOLESKY-Faktor **G** existiert, folgt: Die Matrix **A** ist positiv definit.

3.2.3 FORTRAN-Programm für die CHOLESKY-Zerlegung

FORTRAN-Programme für die CHOLESKY-Zerlegung:

```
...../.....................................
      SUBROUTINE CHOLES(A,NEQ)
      DIM A(NEQ,NEQ)
C.....
      DO 100 J=1,NEQ
        J1 = J-1
        DO 200 K=1,J1
200...    A(J,J) = A(J,J) - A(J,K)*A(J,K)
        IF (A(J,J).LE.0) RETURN
        A(J,J) = SQRT(A(J,J))
        J2 = J+1
        DO 300 I =J2,NEQ
          DO 400 K=1,J1
400         A(I,J) = A(I,J) - A(I,K)*A(J,K)
300     A(I,J) = A(I,J)/A(J,J)
100     CONTINUE
      RETURN
...../.....................................
```

3.2.4 Numerischer Aufwand für die CHOLESKY – Zerlegung

Die Zahl der numerischen Operationen für die CHOLESKY-Zerlegung kann anhand des Algorithmus in Gl. (2.5.11c), (2.5.12c), (2.5.13c) bis (2.5.14c) des FORTRAN-Programms abgeschätzt werden. Für die Berechnung der Nebendiagonalelemente ergibt sich:

g_{i1} aus Gl. (1c):	1 op.	;	für (N-1)-Elemente:	(N-1) op.
g_{i2} aus Gl. (2c):	2 op.	;	für (N-2)-Elemente:	2 (N-2) op.
g_{i3} aus Gl. (3c):	3 op.	;	für (N-3)-Elemente:	3 (N-3) op.
⋮			⋮	⋮
g_{ij} aus Gl. (4c):	j op.	;	für (N-j)-Elemente:	j (N-j) op.
⋮			⋮	⋮
g_{iN-1}	N op.	;	für 1-Element:	N op.

$$\#\,\text{op.} = \sum_{j=1}^{N} j\,(N-j)$$

$$= \sum_{j=1}^{N} j\,N - \sum_{J=1}^{N} J^2$$

$$= \underbrace{N \sum_{j=1}^{N} j} - \underbrace{\sum_{j=1}^{N} j^2}$$

$$\cong N\,\frac{N^2}{2} - \frac{N^3}{3}$$

Summe der Rechenoperationen für die Nebendiagonalglieder:

$$\boxed{\#\,\text{op.} = \frac{1}{6}\,N^3}$$

Das Hauptdiagonalglied g_{jj} benötigt gemäß Gl. (2.5.14b) insgesamt $(j-1)$-Operationen. Für alle N-Hauptdiagonalglieder sind

$$\#\,\text{op.} = \sum_{j=1}^{N} (j-1) \cong \frac{1}{2}\,N^2$$

Operationen erforderlich. Im Vergleich zur Zahl der Rechenoperationen für die Nebendiagonalglieder ist bei großen Gleichungssystemen der numerische Aufwand für die Ermittlung der Hauptdiagonalglieder vernachlässigbar, so dass die Zahl der Rechenoperationen für die CHOLESKY-Zerlegung mit $N^3/6$ Rechenoperationen abgeschätzt werden kann.

3.2.5 Zusammenfassung der FORTRAN – Programme

FORTRAN-Programm zur Vorwärtsreduktion:

```
...../.....................................
      SUBROUTINE SOLTRI(g,b,neq)
      DIM b(neq), g(neq,neq)
C
      DO 100 i  =  1,neq
        if (g(i,i).eq.0) RETURN
        il = i-l
        DO 200 j=l,il
200       b(i) = b(i) - g(i,j)*b(j)
100     b(i) = b(i) / g(i,i)
      RETURN
...../.....................................
```

FORTRAN-Programm für die CHOLESKY-Zerlegung:

```
...../.....................................
      SUBROUTINE choles(a,neq)
      DIM a(neq,neq)
      DO 100 j=1,neq
        jl = j-l
        DO 200 k=l,jl
200       a(j,j) = a(j,j) - a(j,K)*a(j,k)
        IF(a(j,j).le.0)  RETURN
        a(j,j)  =  sqrt(a(j,j))
        j2  =  j+1
        DO 300 i=j2,neq
          DO 400 k=l,jl
400       a(i,j) = a(i,j) - a(i,k)*a(j,k)
300     a(i,j)  =  a(i,j) /a(j,j)
100   CONTINUE
      RETURN
...../.....................................
```

3.2.6 Zum Beweis der Positivität der Pivotelemente

Zur Überprüfung der positiven Definitheit der Koeffizientenmatrix werden die Pivotelemente auf Positivität überprüft, was eine notwendige Bedingung für ein gutgestelltes Randwertproblem eines brauchbaren Systems der linearen Elastostatik ist. Deshalb soll der Beweis für diesen Satz hier nachgeliefert werden.

Beweis zum Satz: Falls $\mathbf{A}$ positiv definit ist, dann sind alle **"Pivotelemente"** positiv.

Beweis: Kern des Beweises ist die Teilung (Partitionierung) der Matrix $\mathbf{A}$ in

$$\mathbf{A} = \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{a} \\ \mathbf{a}^{\mathsf{T}} & \alpha \end{bmatrix},$$

wobei von der HOUSEHOLDER-Schreibweise Gebrauch gemacht wird, dass nämlich

- Großbuchstaben für Matrizen,
- Kleinbuchstaben für Vektoren
- und griechische Buchstaben für Skalare

stehen. Durch Nachrechnen lässt sich zeigen, dass folgende Zerlegung der partitionierten Matrix $\mathbf{A}$ gilt:

$$\mathbf{A} = \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{a} \\ \mathbf{a}^{\mathsf{T}} & \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{a}^{\mathsf{T}}\hat{\mathbf{A}}^{-1} & 1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{0} \\ \mathbf{0}^{\mathsf{T}} & \gamma \end{bmatrix} \begin{bmatrix} \mathbf{I} & \hat{\mathbf{A}}^{-1}\mathbf{a} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix}$$

Nach Ausführen der Multiplikation der Matrizen ergibt sich für

$$\gamma = \alpha - \mathbf{a}^{\mathsf{T}}\hat{\mathbf{A}}^{-1}\mathbf{a}\ , \tag{2.5.15}$$

wobei γ auch als SCHUR-Komplement von $\hat{\mathbf{A}}$ in $\mathbf{A}$ bezeichnet wird. Gleichzeitig ist γ das letzte **Pivotelement** im GAUSSschen Eliminationsverfahren der Matrix $\mathbf{A}$.

Durch Vertauschen von Zeilen und Spalten der Matrix $\mathbf{A}$ kann jedes Pivotelement in die letzte Zeile gebracht werden, so dass γ als repräsentatives Pivotelement für alle anderen Pivotelemente betrachtet werden kann. Das Vertauschen von Zeilen und Spalten ändert nämlich nichts an der Lösbarkeit des Gleichungssystems, sondern nur die Reihenfolge der Unbekannten wird umsortiert.

Zu zeigen ist: $\gamma > 0$

Idee: Suche ein $\mathbf{y} \in \mathbb{R}^n$, so dass gilt:

$$\gamma = \mathbf{y}^T \mathbf{A}\,\mathbf{y}$$

Da **A** nach Voraussetzung positiv definit ist, gilt für jedes $y \in \mathbb{R}^n$

$$\mathbf{x}^T \mathbf{A}\,\mathbf{x} > 0 \ ,$$

also auch für $\mathbf{x} = \mathbf{y}$.

Gl. (2.5.15) kann umgeschrieben werden in die Form:

$$\gamma = \mathbf{a}^T\ \hat{\mathbf{A}}^{-1}\mathbf{a} - 2\,\mathbf{a}^T \hat{\mathbf{A}}^{-1}\mathbf{a} + \alpha$$

$$= \left(-\hat{\mathbf{A}}^{-1}\mathbf{a}\right)^T \hat{\mathbf{A}} \left(-\hat{\mathbf{A}}^{-1}\mathbf{a}\right) + \mathbf{a}^T \left(-\hat{\mathbf{A}}^{-1}\mathbf{a}\right) + \left(-\hat{\mathbf{A}}^{-1}\mathbf{a}\right)^T \mathbf{a} + \alpha$$

$$= \underbrace{\begin{bmatrix} -\hat{\mathbf{A}}^{-1} & \mathbf{a} \\ \mathbf{1} & \end{bmatrix}^T}_{=} \underbrace{\begin{bmatrix} \hat{\mathbf{A}} & \mathbf{a} \\ \mathbf{a}^T & \alpha \end{bmatrix}}_{=} \underbrace{\begin{bmatrix} -\hat{\mathbf{A}}^{-1} & \mathbf{a} \\ \mathbf{1} & \end{bmatrix}}_{=}$$

$$= \quad \mathbf{y}^T \qquad \mathbf{A} \qquad \mathbf{y} \quad > 0, \quad \text{da } \mathbf{A} \text{ positiv definit.}$$

q.e.d.

3.3 Lösung von linearen Gleichungssystemen mittels CHOLESKY – Zerlegung

Wenn ein Gleichungssystem für mehrere "rechte Seiten" gelöst werden muss - d.h. mehrere Lastfälle auftreten, so ist es vorteilhaft, die Koeffizientenmatrix in die CHOLESKY-Faktoren $\mathbf{G}\mathbf{G}^T$ zu zerlegen und danach das Gleichungssystem durch Vorwärtsreduzieren und Rückwärtseinsetzen für die einzelnen Lastfälle zu lösen. Der Rechenaufwand für die Faktorisierung der Koeffizientenmatrix, welche den Großteil der numerischen Operationen ausmacht, muss in diesem Fall nur einmal geleistet werden.

Zusammenfassung des Algorithmus`:

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

CHOLESKY-Zerlegung: $\mathbf{G}\mathbf{G}^T\,\mathbf{x} = \mathbf{b}$, wobei $G_{ij} = 0$, falls $j > i$

Definiere: $\mathbf{y} = \mathbf{G}^T \mathbf{x}$

Löse: $\mathbf{G}\,\mathbf{y} = \mathbf{b}$

durch "Vorwärtselimination (Vorwärtsreduzieren)" für $\mathbf{y}$ - siehe Kapitel 2.5.1.

Löse: $\mathbf{G}^T \mathbf{x} = \mathbf{y}$

durch "Rückwärtseinsetzen" für $\mathbf{x}$.

Anhand des Ergebnisses für die Zerlegung linearer Gleichungssysteme mit dreieckförmig belegter unterer Koeffizientenmatrix in Kapitel 2.5.1 ergibt sich für die Vorwärtsreduktion zur Berechnung des Elements y_i:

$$\boxed{y_i = \frac{1}{g_{ii}} \left(b_i - \sum_{j=1}^{i-1} g_{ij}\, y_j \right)} \qquad i = 1, 2, 3, \ldots, n \qquad (2.5.16)$$

Für das Rückwärtseinsetzen muss ein Gleichungssystem mit oben besetzter dreieckförmiger Koeffizientenmatrix gelöst werden.

$$\boxed{x_i = \frac{1}{g_{ii}} \left(y_i - \sum_{j=i+1}^{n} g_{ji}\, x_j \right)} \qquad i = n, n-1, n-2, \ldots, 1 \qquad (2.5.17)$$

$$\uparrow$$

Auf die Vertauschung der Indizes $(\,)_{ij}$ in den Summanden g_{ij} wird hingewiesen, da hier die transponierte Matrix $\mathbf{G}^T$ steht.

Beginnend mit der Berechnung des letzten Elements für $i = n$, $n-1$, $n-2$, ... usw. ergibt sich aus Gl. (2.5.17):

$$x_n = (y_n - 0) / g_{nn}$$

$$x_{n-1} = (y_{n-1} - g_{n,n-1}\, x_n) / g_{n-1,n-1}$$

$$x_{n-2} = (y_{n-2} - g_{n,n-2}\, x_n - g_{n-1,n-2}\, x_{n-1}) / g_{n-2,n-2}$$

$$\vdots$$

Beispiel: Das Gleichungssystem $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ mit

$$\begin{bmatrix} 4 & -2 & 4 & 2 \\ -2 & 10 & -2 & -7 \\ 4 & -2 & 8 & 4 \\ 2 & -7 & 4 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \\ 16 \\ 6 \end{bmatrix}$$

soll anhand des CHOLESKY-Faktors **G** (siehe Beispiel in Kapitel 2.5.2) durch Vorwärtsreduktion und Rückwärtseinsetzen gelöst werden.

Zu lösen ist die Gleichung

$$\mathbf{G}\mathbf{y} = \mathbf{b}$$

d.h.

$$\begin{bmatrix} 2 & & & \\ -1 & 3 & & \\ 2 & 0 & 2 & \\ 1 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \\ 16 \\ 6 \end{bmatrix}$$

durch Vorwärtsreduktion mit Hilfe von Gl. (2.5.16):

$$y_1 = 8/2 = 4$$

$$y_2 = \left[2-(-1\cdot 4)\right]/3 = 2$$

$$y_3 = \left[16-(2\cdot 4+0\cdot 2)\right]/2 = 4$$

$$y_4 = \left[6-(1\cdot 4+(-2)\,2+1\cdot 4)\right]/1 = 2$$

Danach ist die Beziehung

$$\mathbf{G}^{\mathrm{T}}\,\mathbf{x} = \mathbf{y}$$

d.h.
$$\begin{bmatrix} 2 & -1 & 2 & 1 \\ & 3 & 0 & -2 \\ & & 2 & 1 \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 4 \\ 2 \end{bmatrix}$$

durch Rückwärtseinsetzen mit Hilfe von Gl. (2) zu lösen. Sie liefert die Lösung des linearen Gleichungssystems.

$$x_4 = 2/1 \qquad = 2$$

$$x_3 = \left[4 - (1 \cdot 2)\right]/2 \qquad = 1$$

$$x_2 = \left[2 - (-2 \cdot 2 + 0 \cdot 1)\right]/3 \qquad = 2$$

$$x_1 = \left[4 - (1 \cdot 2 + 2 \cdot 1 + (-1)2)\right]/2 \qquad = 1$$

Der Lösungsvektor lautet also:

$$\mathrm{x} = \left[1 \;\; 2 \;\; 1 \;\; 2\right]^T .$$

Die Gesamtzahl der numerischen Operationen für die CHOLESKY-Zerlegung, die Vorwärtsreduktion und das Rückwärtseinsetzen ergibt sich zu:

$$\#\,\mathrm{op.} = \frac{1}{6} N^3 + \frac{1}{2} N^2 + \frac{1}{2} N^2$$

Die Zahl der Rechenoperationen für Vorwärtsreduktion und Rückwärtseinsetzen ist jedoch im Vergleich zum Rechenaufwand für die CHOLESKY-Zerlegung bei großen Gleichungssystemen vernachlässigbar.

Ein FORTRAN-Programm zur Gleichungslösung mit Hilfe des CHOLESKY-Faktors kann wie folgt aussehen:

```
.........!.........................................

SUBROUTINE SOLVEB(A,B,NEQ)
C .....
C .....A... CHOLESKY-FAKTOR  G
C .....B... RIGHT HAND SIDE
      DIMENSION B(NEQ), A(NEQ,NEQ)
```

```
      DO 100 I=2,NEQ
        I1=I-2
        DO 100 K=1,I1
100       B(I) = B(I) - A(I,K)*B(K)
      I=NEQ
400   B(I) = B(I)/A(I,I)
      IF (I.EQ.NEQ) GO TO 300
      II = I+1
      DO 200 K=II,NEQ
200       B(I) = B(I) - A(K,I)*B(K)
300     I = I - 1
      IF (I.NE.0) GO TO 400
      RETURN
      ...../.....................................
```

Hinweis: Der Lösungsvektor **x** steht am Ende auf dem Feld **B**, wo anfangs die rechte Seite abgespeichert war.

Ein verkürztes FORTRAN-Programm zur Lösung des Gleichungssystems kann wie folgt lauten:

```
      ....../.....................................
      SUBROUTINE SOLVEB(A,B,NEQ)
C....
C.... A... CHOLESKY-FACTOR G
C.... B... RIGHT HAND SIDE
      DIMENSION B(NEQ), A(NEQ,NEQ)
      DO 100 I=1,NEQ
         DO 100 K=1,I-1
 100        B(I) = B(I) - A(I,K)*B(K)
      DO 200 I=NEQ,1,-1
      B(I) = B(I)/A(I,I)
      DO 300 K=I+1,NEQ
 300     B(I) = B(I) - A(K,I)*B(K)
 200  B(I) = B(I)/A(I,I)
      RETURN
      END
```

3.4 Gleichungslösung mittels GAUSSscher Elimination und LU - Zerlegung

Zur Lösung des linearen Gleichungssystems durch das GAUSSsche Eliminationsverfahren werden nur elementare Zeilenoperationen benutzt. Im Gegensatz zum CHOLESKY-Verfahren können auch Gleichungssysteme mit nicht positiv definiter Koeffizientenmatrix oder unsymmetrischer Koeffizientenmatrix bearbeitet werden.

3.4.1 Demonstration des GAUSSschen Verfahrens am Zahlenbeispiel

Beispiel: 3 Gleichungen mit 3 Unbekannten

$$5\,x_1 + 4\,x_2 + 3\,x_3 = 2{,}0 \tag{2.5.18}$$

$$4\,x_1 + 7\,x_2 + 4\,x_3 = -1{,}0 \tag{2.5.19}$$

$$3\,x_1 + 4\,x_2 + 4\,x_3 = 3{,}0 \tag{2.5.20}$$

$$5\,x_1 + 4\,x_2 + 3\,x_3 = 2{,}0 \tag{2.5.18}$$

$(2.5.19) - \frac{4}{5}(2.5.18)$: $\quad 3{,}8\,x_2 + 1{,}6\,x_3 = -2{,}6$ (2.5.21)

$(2.5.20) - \frac{3}{5}(2.5.18)$: $\quad 1{,}6\,x_2 + 2{,}2\,x_3 = 1{,}8$

(2.5.22)

$$5\,x_1 + 4\,x_2 + 3\,x_3 = 2{,}0 \tag{2.5.18}$$

$$3{,}8\,x_2 + 1{,}6\,x_3 = -2{,}6 \tag{2.5.22}$$

$(2.5.22) - \underbrace{\frac{1{,}6}{3{,}8}}_{=0{,}421}(2.5.21)$ $\quad 1{,}527\,x_3 = 2.894$ (2.5.23)

$$\underbrace{2{,}2 - \frac{1{,}6}{3{,}8}1{,}6}\quad \underbrace{1{,}8 - \frac{1{,}6}{3{,}8}(-2{,}6)}$$

Das Gleichungssystem in (2.5.18) bis (2.5.23) mit oben besetzter dreiecksförmiger Koeffizientenmatrix kann durch Rückwärtseinsetzen gelöst werden.

Die Multiplikatoren $m_{21} = \frac{4}{5} = 0{,}8$, $m_{31} = \frac{3}{5} = 0{,}6$ und $m_{32} = \frac{1{,}6}{3{,}8} = 0{,}421$ werden zusammen mit den Koeffizienten der oberen Dreiecksmatrix **U** in einer Matrix gespeichert.

$$\begin{bmatrix} 5{,}000 & 4{,}000 & 3{,}000 \\ 0{,}800 & 3{,}800 & 1{,}600 \\ 0{,}600 & 0{,}421 & 1{,}527 \end{bmatrix}$$

Muss das ursprüngliche Gleichungssystem $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ für eine neue rechte Seite nochmals gelöst werden, so bleibt die obere Dreiecksmatrix **U** unverändert. Nur elementare Zeilenoperationen müssen mit Hilfe der gespeicherten Multiplikatoren in der unteren Dreiecksmatrix am Spaltenvektor **b** vorgenommen werden.

Beispiel: GAUSSsche Elimination

$$\begin{bmatrix} 5 & 4 & 3 \\ 4 & 7 & 4 \\ 3 & 4 & 4 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad = \quad \begin{bmatrix} 2{,}0 \\ -1{,}0 \\ 3{,}0 \end{bmatrix}$$

A **x** = **b**

$m_{21} = 4/5 = 0{,}8$
$m_{31} = 3/5 = 0{,}6$

$$\begin{bmatrix} 5 & 4 & 3 \\ & 3{,}8 & 1{,}6 \\ & 1{,}6 & 2{,}2 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2{,}0 \\ -1{,}0 \\ 3{,}0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0{,}8 \cdot 2{,}0 \\ 0{,}6 \cdot 2{,}0 \end{bmatrix} = \begin{bmatrix} 2{,}0 \\ -2{,}6 \\ 1{,}8 \end{bmatrix}$$

$$\ldots \qquad = \underbrace{[b_i \;] \quad - [m_{i1}\, b_1]}_{\left[b_i^{(1)}\right]}$$

$m_{32} = 1{,}6/3{,}8 = 0{,}421$

$$\begin{bmatrix} 5 & 4 & 3 \\ 0 & 3{,}8 & 1{,}6 \\ 0 & 0 & 1.527 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2{,}0 \\ -1{,}0 \\ 3{,}0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0{,}8 \cdot 2{,}0 \\ 0{,}6 \cdot 2{,}0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0{,}421 \cdot (-2{,}6) \end{bmatrix} = \begin{bmatrix} 2{,}0 \\ -2{,}6 \\ 2{,}894 \end{bmatrix}$$

$$\begin{bmatrix} \quad u_{ij} \quad \end{bmatrix} \begin{bmatrix} x_j \end{bmatrix} = \underbrace{\begin{bmatrix} \; b_i \; \end{bmatrix} - \begin{bmatrix} m_{i1}\, b_i \end{bmatrix}}_{\left[b_i^{(1)}\right]} - \begin{bmatrix} m_{i2}\, b_2^{(1)} \end{bmatrix}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\left[b_i^{(2)}\right]}$$

In verkürzter Schreibweise ergibt sich für den Vektor $\mathbf{b} = [2 \quad -1 \quad 3]^T$ aus vorigem Beispiel:

$$b_2^{(1)} = b_2 - m_{21}\, b_1 \qquad = -1{,}0 - 0{,}8 \cdot 2{,}0 \qquad = -2{,}6$$

$$b_3^{(1)} = b_3 - m_{31}\, b_1 \qquad = 3{,}0 - 0{,}6 \cdot 2{,}0 \qquad = 1{,}8$$

$$b_3^{(2)} = b_3^{(1)} - m_{32}\, b_2^{(1)} \qquad = 1{,}8 - 0{,}421 \cdot (-2{,}6) = 2{,}894$$

3.4.2 GAUSSscher Algorithmus

Satz: Jede $n \times n$-Matrix **A**, deren führende Hauptuntermatrizen nicht singulär sind, lässt sich mit Hilfe des GAUSSschen Eliminationsverfahrens in eine obere Dreiecksmatrix **U** durch elementare Zeilenoperationen umformen.

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ m_{21} & u_{22} & u_{23} & \dots & u_{2n} \\ m_{31} & m_{32} & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & & \\ m_{n1} & m_{n2} & m_{n3} & \dots & u_{mn} \end{bmatrix}$$

Die rechte Seite **b** wird mit gegebenen Komponenten b_i wie folgt umgeformt:

$$\begin{aligned}
b_i^{(1)} &= b_i - m_{i1}\, b_1 && i = 2, 3, 4, \ \dots\ j,\ j+1,\ \dots n \\
b_i^{(2)} &= b_i^{(1)} - m_{i2}\, b_2^{(1)} && i = \quad 3, 4, \ \dots\ j,\ j+1, \dots\ n \\
b_i^{(3)} &= b_i^{(2)} - m_{i3}\, b_3^{(2)} && i = \qquad 4, \ \dots j,\ j+1, \dots n \\
&\ \vdots \qquad \vdots && \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (2.5.24) \\
b_i^{(j)} &= b_i^{(j-1)} - m_{ij}\, b_j^{(j-1)} && i = \qquad\qquad\quad j+1, \dots, n \\
&\ \vdots \\
b_i^{(n-1)} &= b_i^{(n-2)} - m_{i,n-1}\, b_{n-1}^{(n-2)} && i = \qquad\qquad\qquad\quad n
\end{aligned}$$

Am Ende dieser Umformungen geht **b** in den Vektor **y** über und lautet:

$$\mathrm{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(n-1)} \end{bmatrix} \tag{2.5.25}$$

Bei näherer Betrachtung der Umformungen von **b** erkennt man, dass die Komponenten des Vektors **y** durch Rekursion berechnet werden können. Mit Gl. (2.5.25) ergibt sich aus (2.5.24):

$$b_i^{(1)} = b_i - m_{ij}\, y_1 \qquad i = 2, 3, 4, \ldots, n \tag{2.5.26}$$

$$b_i^{(2)} = b_i^{(1)} - m_{i2}\, y_2 \qquad i = 3, 4, \ldots, n \tag{2.5.27}$$

$$b_i^{(3)} = b_i^{(2)} - m_{i3}\, y_3 \qquad i = 4, \ldots, n \tag{2.5.28}$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$b_i^{(n-1)} = b_i^{(n-2)} - m_{i,n-1}\, y_{n-1} \qquad i = n \tag{2.5.29}$$

Aus Gl. (2.5.26) bis (2.5.29) folgt durch wiederholtes Anwenden der Beziehung (2.5.26) bis (2.5.29):

$$y_1 = b_1$$

$$y_2 = b_2^{(1)} \overset{(2.5.26)}{=} b_2 - m_{21}\, y_1 = b_2 - \sum_{j=1}^{1} m_{2j}\, y_j \tag{2.5.30}$$

$$y_3 = b_3^{(2)} \overset{(2.5.27)}{=} b_3^{(1)} - m_{32}\, y_2 \overset{(2.5.26)}{=} b_3 - m_{31}\, y_1 - m_{32}\, y_2 = b_3 - \sum_{j=1}^{2} m_{3j}\, y_j \tag{2.5.31}$$

$$y_4 = b_4^{(3)} \overset{(2.5.28)}{=} b_4^{(2)} - m_{43}\, y_3 \overset{(2.5.27)}{=} b_4^{(1)} - m_{42}\, y_2 - m_{43}\, y_3 \overset{(4.1)}{=} b_4 - \sum_{j=1}^{3} m_{4j}\, y_j \tag{2.5.32}$$

$$\vdots$$

$$y_i = b_i^{(i-1)} \overset{(4.i)}{=} \ldots = b_i - \sum_{j=1}^{i-1} m_{ij}\, y_j \qquad i = 1, 2, \ldots, n \tag{2.5.33}$$

Gl. (5.i) kann umgestellt und auf die Form eines linearen Gleichungssystems gebracht werden.

$$\sum_{j=1}^{i-1} m_{ij}\ y_j\ +\ y_i = b_i \qquad\qquad i = 1, 2, \ldots, n \qquad\qquad (2.5.34)$$

Als Matrizengleichung lautet es:

$$\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ m_{21} & 1 & 0 & \ldots & 0 \\ m_{31} & m_{32} & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ m_{n1} & m_{n2} & m_{n3} & \ldots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Der Vektor $\mathbf{y}$ ist Lösung des linearen Gleichungssystems $\mathbf{Ly} = \mathbf{b}$, wobei $\mathbf{L}$ eine unten besetzte Dreiecksmatrix ist.

Zusammenfassend kann festgehalten werden, dass das System

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \qquad\qquad (2.5.35)$$

auf die Form

$$\mathbf{Ux} = \mathbf{y} \qquad\qquad (2.5.36)$$

reduziert werden kann, wobei sich $\mathbf{y}$ aus

$$\mathbf{Ly} = \mathbf{b} \qquad\qquad (2.5.37)$$

berechnet. Einsetzen von Gl. (8) in (9) liefert:

$$\mathbf{LUx} = \mathbf{b} \qquad\qquad (2.5.38)$$

Gl. (2.5.35) und (2.5.38) gelten für alle Vektoren $\mathbf{x}$, so dass $\mathbf{A}$ in das Produkt von zwei Dreiecksmatrizen

$$\mathbf{A} = \mathbf{LU} \qquad\qquad (2.5.39)$$

zerlegt werden kann. Es muss noch gezeigt werden, dass die Zerlegung eindeutig ist.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{31} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix} \tag{2.5.40}$$

Die Elemente der ersten Zeile von **U** sind identisch mit denjenigen der ersten Zeile von **A**, also

$$u_{ij} = a_{ij}. \qquad i = 1,2,3,...,n$$

Aus den Produkten der i-ten Zeile von **L** mit der ersten Spalte von **U** ergibt sich:

$$l_{i1} = \frac{a_{i1}}{u_{11}} \qquad i = 2, 3, 4,..., n$$

Damit ist die erste Spalte von **L** bestimmt. Aus dem Produkt der zweiten Zeile von **L** mit den Spalten von **U** ergeben sich die Elemente der zweiten Zeile von **U** entsprechend der Beziehung:

$$a_{2j} = l_{21}\, u_{1j} + 1 \cdot u_{2j}$$

$$\Leftrightarrow \qquad u_{2j} = a_{2j} - l_{21}\, u_{1j} \qquad j = k, k+1,..., n$$

Desweiteren folgt:

$$l_{j2} = \left(a_{j2} - l_{j1} u_{12}\right) / u_{22}$$

Unter der Annahme, dass die ersten $(k-1)$-Zeilen von **U** und Spalten von **L** bekannt sind, kann das Element u_{kj} aus dem Produkt der Zeile k aus der Matrix **L** mit der Spalte j aus der Matrix **U** bestimmt werden.

$$a_{kj} = \sum_{m=1}^{k-1} l_{km}\, u_{mj} + 1 \cdot u_{kj} \qquad j = k, k+1,..., n$$

$$\Leftrightarrow \qquad \boxed{u_{kj} = a_{kj} - \sum_{m=1}^{k-1} l_{km}\, u_{mj}} \qquad j = k, k+1,..., n \tag{2.5.41}$$

Da u_{kk} bekannt ist, können alle Elemente l_{ik} der k-ten Spalte von **L** aus dem Produkt der i-ten Zeile von **L** mit der k-ten Spalte von **U** berechnet werden.

$$a_{ik} = \sum_{m=1}^{h-1} l_{im}\, u_{mk} + l_{ik}\, u_{kk}$$

$$\Leftrightarrow \qquad \boxed{l_{ik} = \frac{a_{ik} - \sum_{m=1}^{k-1} l_{im}\, u_{mk}}{u_{kk}}} \qquad i = k, k+1, \ldots, n \qquad (2.5.42)$$

Damit ist die Eindeutigkeit der **LU**-Zerlegung von **A** durch vollständige Induktion bewiesen. Man nennt die in Gl. (2.5.41) und (2.5.42) beschriebene Form der **LU**-Zerlegung die Skalarproduktform oder DOLITTLE-Reduktion. Sie ist eine Variante der CROUT-Reduktion. Im Unterschied zur Skalarproduktform der **LU**-Zerlegung werden beim GAUSSschen Eliminationsverfahren die Elemente von **U** viele Male verändert.

Eine wichtige Variante der **LU**-Zerlegung ist die **LDV**-Zerlegung, wo eine Diagonalmatrix **D** zwischen zwei Dreiecksmatrizen **L** und **V** steht, die auf der Hauptdiagonalen nur Elemente der Größe 1 haben.

$$\mathbf{A} = \mathbf{LDV}$$

Ist **A** eine symmetrische Matrix, deren führende Hauptuntermatrizen nicht singulär sind, so kann **A** in das Produkt

$$\mathbf{A} = \mathbf{LDL}^{\mathrm{T}}$$

zerlegt werden, wobei **L** eine untere Dreiecksmatrix mit Einsen auf der Hauptdiagonalen und **D** eine Diagonalmatrix ist, deren Nebendiagonalglieder alle null sind.

3.4.3 FORTRAN-Subroutine zum GAUSSschen Lösungsverfahren

Die Skalarproduktform der LU-Zerlegung einer symmetrischen Matrix ist im Folgenden als FORTRAN-Subroutine SYMSOL programmiert. Darüberhinaus ist noch die Lösung eines Gleichungssystems für verschiedene rechte Seiten mit Hilfe der LDL^T-Zerlegung codiert worden.

Subroutine SYMSOL (A, B, N, LD)

```
       SUBROUTINE SYMSOL (A,B.N,LD)
       DIMENSION A(N,N). B(N,LD)
C #####SOLUTION BY LDLT FACTORIZATION      ################
C #####INNER PRODUCT VERSION OF LDL^T
       IF (N.EQ.1)  GO TO 500
       DO  400  J=2,N
         JJ=J-1
C---------------------------------------         U (I,J)
         DO  300  I=1,JJ
           II=I-1
           IF (II.EQ.0)  GO TO  300
           SUM=0.0
           DO  250  K=1,II
   250       SUM=SUM+A(K,I) * A(K,J)
           A(I,J) = A(I,J)-SUM
   300   CONTINUE
C---------------------------------------         D (J,J) AND L (I,J)
         SUM=0.0
         DO 350  K=1,JJ
           TEMP=A(K,J) / A(K,K)
           SUM=SUM+TEMP*A (K,J)
   350     A(K,J)=TEMP
         A(J,J) = A(J,J)-SUM
C---------------------------------------
   400   CONTINUE
C ####   SOLVE FOR ALL LOAD CONDITIONS     #####
   500   DO 700 L=1, LD
C---------------------------------------         Y (I,L)
         IF (N.EQ.1)  GO TO  600
         DO 550  I=2, N
         II=I-1
         DO 550  K=1, II
```

```
   550   B(I,L) = B(I,L)-A(K,I)*B(K,L)
C---------------------------------------          X (I,L)
   600   I=N
   650   B(I,L)=B(I,L)-A(I,I)
         IF (I.EQ.N) GO TO 690
         II=I+1
         DO 675 K=II,N
   675   B(I,L)=B(I,L)-A(I,K)*B(K,L)
   690   I=I-1
         IF (I.NE.0) GO TO 650
C---------------------------------------
   700   CONTINUE
C ##########################
         RETURN
         END
```

4 FEM für Stabsysteme (Fachwerke und Rahmen)

Stabsysteme sind Rahmen mit biege- und dehnweichen Stäben. Im Sonderfall von gelenkig verbundenen Stäben liegt ein Fachwerk vor.

4.1 Grundgleichungen des Balkens

Die allgemeine Beanspruchung eines Stabes besteht aus Längsdehnung, Biegung und Querkraft. Man spricht dann von einem Balken. Durch Angabe von Anfangs- und Endknoten wird die Orientierung des Balkenelements festgelegt. Daraus bestimmt sich das Vorzeichen für die Schnittgrößenberechnung in einem finiten Elemente-Programm.

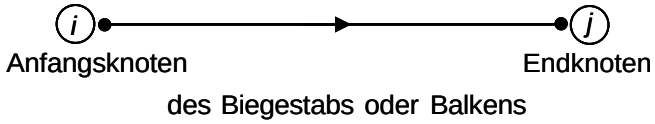


Abb. 4.1-1: Orientierung des Balkenelements zur Schnittgrößendefinition

Die Modellbildung für mechanische Aufgaben lässt sich mit drei Gleichungstypen beschreiben.

4.1.1 Gleichgewicht am Biegestab

Bestimmung des Gleichgewichts zwischen den äußeren Lasten $n(x)$ und $q(x)$ sowie den Schnittgrößen $N(x), Q(x)$ und $M(x)$.

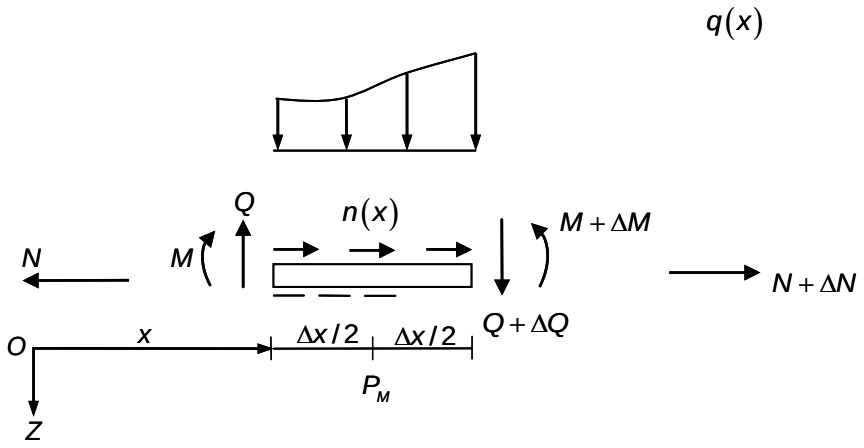


Abb. 4.1-2: Balkenelement mit Schnittgrößen

Kräftegleichgewicht:

$$\sum F_{iz} = 0 = -Q + q(x)\Delta x + (Q + \Delta Q)$$
$$\frac{\Delta Q}{\Delta x} = -q(x)$$

$$\boxed{Q' := \lim_{\Delta x \to 0} \frac{\Delta Q}{\Delta x} = -q(x)}$$

Momentengleichgewicht:

$$\sum M_{i\,P_M = P\left(x + \frac{\Delta x}{2}\right)} = -M - Q\frac{\Delta x}{2} - (Q + \Delta Q)\frac{\Delta x}{2} + (M + \Delta M) = 0$$
$$\Delta M = \left(Q\frac{\Delta x}{2}\right)2 + \Delta Q\frac{\Delta x}{2}$$
$$\boxed{M' = \lim_{\Delta x \to 0} \frac{\Delta M}{\Delta x} = Q(x)}$$

Kräftegleichgewicht:

$$\sum F_{ix} = 0 = (N + \Delta N) - N + n(x)\Delta x$$
$$\boxed{N' = \lim_{\Delta x \to 0} \frac{\Delta N}{\Delta x} = -n(x)}$$

4.1.2 Kinematik des Balkens

Die Differentialgleichung zwischen den Dehnungen ε und Krümmungen κ sind in Abhängigkeit des Verschiebungsfeldes $u(x,y)$ angegeben.

i) Dehnung (infolge Längslast n(x))

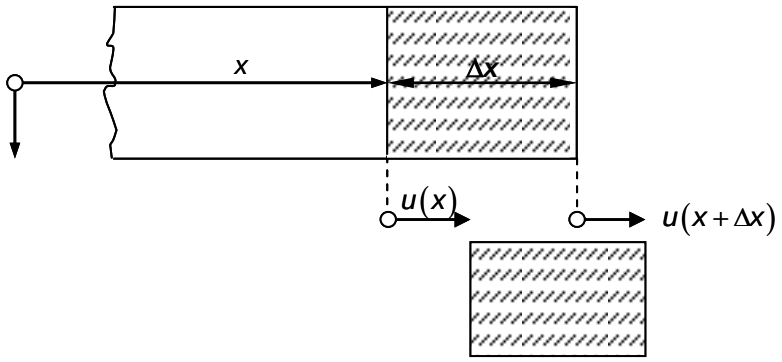


Abb. 4.1-3: Lokale Verformung des Dehnstabs

Am Balkenelement der endlichen Länge Δx gilt:

mittlere Dehnung: $$\bar{\varepsilon} = \frac{u(x+\Delta x) - u(x)}{\Delta x}$$

Grenzübergang $\Delta x \to 0$ von der mittleren Dehnung $\bar{\varepsilon}$ zur

lokalen Dehnung: $$\boxed{\varepsilon = \lim_{\Delta x \to 0} \bar{\varepsilon} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x) - u(x)}{\Delta x} = u'(x)}$$

ii) Biegung mit Querkraft (infolge Querlast q(x))

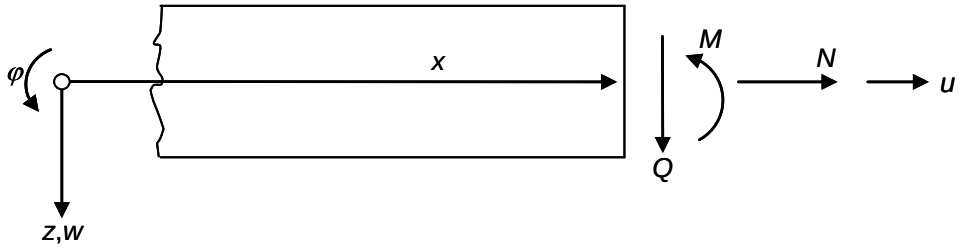


Abb. 4.1-4: Schnittgrößen N,Q und M am Biegestab und Verschiebungen u,w sowie Querschnittsverdrehungen φ

1. Annahme: Querschnitt bleibt eben

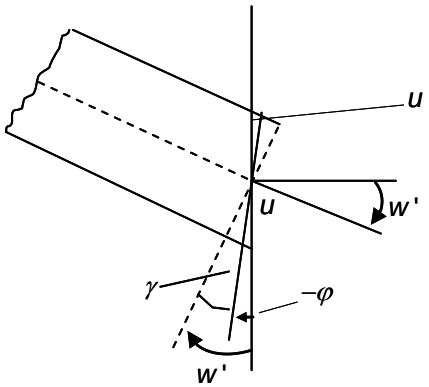


Abb. 4.1-4: Lokale Verformung des Biegestabs

2. Annahme: Querschnitt $\perp$ Schwerlinie

$$\boxed{w' = -\varphi + \gamma}$$

$$\gamma \overset{!}{=} 0$$

Normalenhypothese

$$\varphi = -w'$$

Definition der Krümmung:

$$\kappa = -w'' = +\varphi'$$

$$\boxed{u(x,z) = +\varphi(x)z}$$

Es gibt unterschiedlich genaue Modelle (Theorien) zur Beschreibung der Verformung des Biegebalkens

I) EULER-BERNOULLI-Balkentheorie (Normalenhypothese wird gefordert)

$$\left.\begin{aligned} w' &= -\varphi \\ u &= \varphi z \end{aligned}\right\} \quad \Rightarrow \quad u(x,z) = -w'(x)z$$

Kinematische Gleichungen für das Balkenmodell von L. EULER und J. BERNOULLI:

$$\boxed{\begin{aligned} \varepsilon_x &= \frac{\partial u}{\partial x} = -w''(x)z \\ \gamma_{xz} &= 0 \end{aligned}}$$

Krümmungsverlauf: $\kappa(x) := -w''(x)$

II) TIMOSHENKO - Balkentheorie des schubweichen Balkens

Kinematische Gleichungen für die Balkentheorie von S. TIMOSHENKO:

$$\boxed{\begin{aligned} \varepsilon_x &= \frac{\partial u}{\partial x} = \varphi'(x)z \\ \gamma_{xz} &= \gamma(x) \overset{(5)}{=} w' + \varphi \end{aligned}}$$

4.1.3 Elastizitätsbeziehungen

Im Fall der linearen Elastizitätstheorie gilt:

Normalbeanspruchung:

$$\sigma = E\varepsilon$$

Schubbeanspruchung:

$$\tau = G\gamma$$

i) Elastizitätsbeziehung für die Stabdehnung:

$$\sigma = E\varepsilon_0$$

$$N = \int_A \sigma\, dA = \int_A E\varepsilon_0\, dA = E\varepsilon_0 \underbrace{\int_A dA}_{=A} = EA\varepsilon_0$$

$$\boxed{N = EA\,\varepsilon_0(x)}$$

ii) Elastizitätsbeziehung für die Stabbiegung:

$$M(x) = \int_{(A)} \sigma_x\, z\, dA = \int_{(A)} E\varepsilon_x\, z\, dA = E\int_{(A)} \left[-w''(x)z\right] z\, dA$$

$$= -Ew'' \underbrace{\int_{(A)} z^2\, dA}_{=:I}$$

Flächenträgheitsmoment: $I := \int_{(A)} z^2\, dA$

Einfügen der Krümmung:

$$\boxed{M(x) = +EI\,\kappa(x)}$$

iii) für den Schub:

$$Q = \int_A \tau_{xz}\, dA = \int_A G\gamma_{xz}(x,z)\, dA$$

↑

z.B. parabelförmiger Verlauf über den Rechteckquerschnitt bei konstanter Querkraft **Q**(**x**)

$$\boxed{Q = GA_s\,\gamma(x)}$$

↑↑

„mittlere Schubverformung“ über den Querschnitt

Schubfläche: z.B. Rechteckquerschnitt:

$$A_S = \frac{5}{6}A$$

4.2 Fachwerke

Alle Stäbe sind durch Momentengelenke miteinander verbunden und nehmen keine Biegemomente auf. Es treten nur Knotenlasten auf und keine Einzellasten im Feld eines Einzelstabs. Somit folgt, dass alle Stäbe nur auf Längsdehnung beansprucht werden.

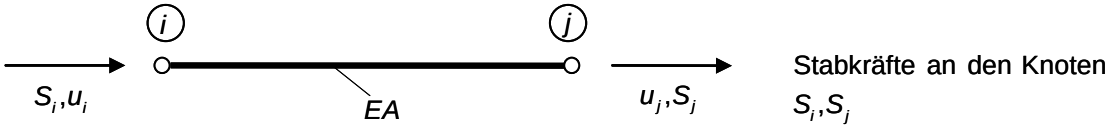


Abb. 4.2-1: Fachwerkelement mit Stablängsverschiebungen und Stabkräften

Die diskretisierte Variationsgleichung des **Dehnstabs** (siehe Skriptum FEM, Kap. 5.1.3) d.h. die schwache Form der Verschiebungsgleichung liefert die lokale Steifigkeitsmatrix **k** im Elementkoordinatensystem.

$$\underbrace{\begin{bmatrix} \frac{EA}{l} & -\frac{EA}{l} \\ -\frac{EA}{l} & \frac{EA}{l} \end{bmatrix}}_{\mathbf{k}} \begin{bmatrix} u_i \\ u_j \end{bmatrix} = \begin{bmatrix} S_i \\ S_j \end{bmatrix}$$

$$\mathbf{k}\,\mathbf{u} = \mathbf{S}$$

Räumliches Fachwerkelment mit Elementfreiheitsgrade v_1 bis v_6

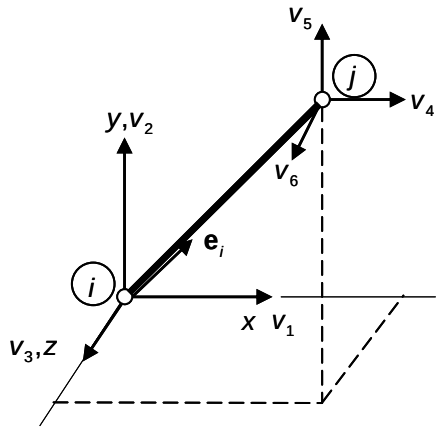


Abb. 4.2-2: Fachwerkelement im Raum

In Richtung der Stabachse wird der Einheitsvektor $\mathbf{e}_i$ eingeführt.

Zerlegung des Vektors der Axialverschiebungen $\mathbf{u}_i = u_i\,\mathbf{e}_i$ und $\mathbf{u}_j = u_j\,\mathbf{e}_j$ an den Enden (i) und (j) des Dehnstabs:

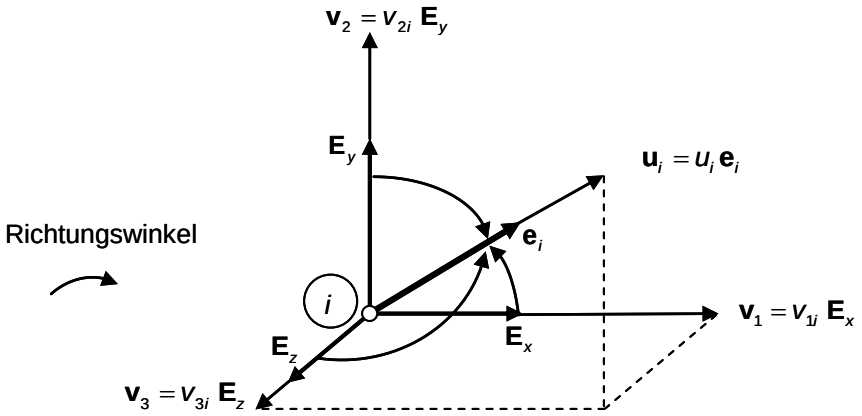


Abb. 4.2-3: Einheitsbasissysteme und Freiheitsgrade am Knoten (*i*)

In symbolischer Notation:

$$\mathbf{u}_i = \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3$$

In Komponentenschreibweise

$$u_i\ \mathbf{e}_i = v_{1i}\ \mathbf{E}_x + v_{2i}\ \mathbf{E}_y + v_{3i}\ \mathbf{E}_z$$

Multiplikation mit dem Einheitsvektor $\mathbf{e}_i$:

$$u_i = v_{1i}\ \mathbf{e}_i \cdot \mathbf{E}_x + v_{2i}\ \mathbf{e}_i \cdot \mathbf{E}_y + v_{3i}\ \mathbf{e}_i \cdot \mathbf{E}_z$$

$$u_i =: v_{1i}\cos\left(\mathbf{e}_i, \mathbf{E}_x\right) + v_{2i}\cos\left(\mathbf{e}_i, \mathbf{E}_y\right) + v_{3i}\cos\left(\mathbf{e}_i, \mathbf{E}_z\right)$$

Analog am Knoten (**j**)

In Matrizennotation mit **a** als Transformationsmatrix für die Verschiebungsfreiheitsgrade:

$$\mathbf{u} = \mathbf{a}\ \mathbf{v}$$

$$\begin{bmatrix} u_i \\ u_j \end{bmatrix} = \begin{bmatrix} c_x & c_y & c_z & 0 & 0 & 0 \\ 0 & 0 & 0 & c_x & c_y & c_z \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}$$

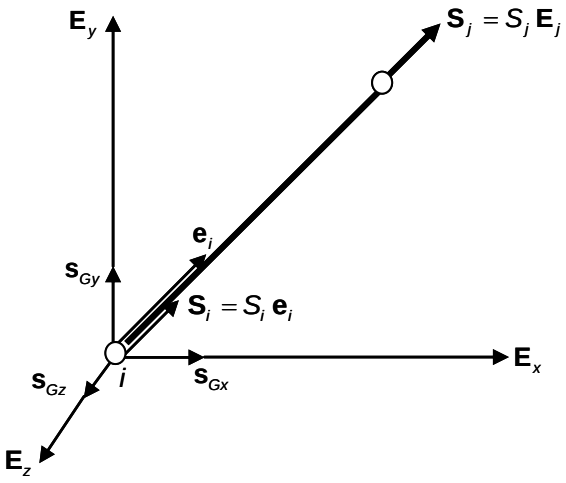
wobei folgende Abkürzungen für die Richtungscosinus gelten sollen:

$$c_x := \cos\left(\mathbf{e}_i, \mathbf{E}_x\right)$$

$$c_y := \cos\left(\mathbf{e}_i, \mathbf{E}_y\right)$$

$$c_z := \cos\left(\mathbf{e}_i, \mathbf{E}_z\right)$$

Zerlegung der Stabkraft $\mathbf{S} = S_i\ \mathbf{e}_i$ in Knotenkräfte $\mathbf{S}_{Gx} = S_{Gx}\mathbf{E}_x$, $\mathbf{S}_{Gy} = S_{Gy}\mathbf{E}_y$ und $\mathbf{S}_{Gz} = S_{Gz}\mathbf{E}_z$



Es gilt :

$$\mathbf{E}_x \cdot \mathbf{E}_y = \mathbf{E}_x \cdot \mathbf{E}_z = \mathbf{E}_y \cdot \mathbf{E}_z = 0$$

$$\mathbf{E}_x \cdot \mathbf{E}_x = \mathbf{E}_y \cdot \mathbf{E}_y = \mathbf{E}_z \cdot \mathbf{E}_z = 1$$

Abb. 4.2-4: Lokale Stabkraft $\mathbf{s}_i$ bzw. $\mathbf{s}_j$ und globale Komponenten

$$\mathbf{S}_i = \mathbf{S}_{Gx} + \mathbf{S}_{Gy} + \mathbf{S}_{Gz}$$

$$\Rightarrow S_i\mathbf{e} = S_{Gx}\,\mathbf{E}_x + S_{Gy}\,\mathbf{E}_y + S_{Gz}\,\mathbf{E}_z$$

Multiplikation der Vektorgleichung für die Stabkraft mit der Einheitsbasis $\mathbf{e}_i$ zum Skalarprodukt ergibt die Richtungskosinus:

$$S_{Gxi} = S_i\ \mathbf{e}_i \square \mathbf{E}_x = S_i \cos(\mathbf{E}_x, \mathbf{e}_i)$$

$$S_{Gyi} = S_i\ \mathbf{e}_i \square \mathbf{E}_y = S_i \cos(\mathbf{E}_y, \mathbf{e}_i)$$

$$S_{Gzi} = S_i\ \mathbf{e}_i \square \mathbf{E}_z = S_i \cos(\mathbf{E}_z, \mathbf{e}_i)$$

Analog werden die Knotenkräfte am Knoten (j) berechnet.

Darstellung der Beziehung zwischen den Komponenten der Knotenkräfte
im Basissystem für die Struktur und den Stabkräften in Matrizenform:

$$\begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{bmatrix} = \begin{bmatrix} S_{Gxi} \\ S_{Gyi} \\ S_{Gzi} \\ S_{Gxj} \\ S_{Gyj} \\ S_{Gzj} \end{bmatrix} = \begin{bmatrix} c_x & 0 \\ c_y & 0 \\ c_z & 0 \\ 0 & c_x \\ 0 & c_y \\ 0 & c_z \end{bmatrix} \begin{bmatrix} S_i \\ S_j \end{bmatrix}$$

$$\mathbf{S_G} = \qquad \mathbf{a^T} \qquad \mathbf{S}$$

Transformation der Elementsteifigkeitsmatrix $\mathbf{k}$ vom lokalen in das globale Koordinatensystem $\mathbf{k}_G$:

$$\mathbf{S}_G = \mathbf{a}^T\ \mathbf{S} = \mathbf{a}^T\mathbf{k}\ \mathbf{u} = \mathbf{a}^T\mathbf{k}\mathbf{a}\ \mathbf{v} =: \mathbf{k}_G\ \mathbf{v}$$

$$\mathbf{k}_G = \mathbf{a}^T\mathbf{k}\mathbf{a}$$

Ausmultiplikation mit dem FALKschen Anordnungsschema:

	$\mathbf{k}$	$\begin{bmatrix} c_x & c_y & c_z & 0 & 0 & 0 \\ 0 & 0 & 0 & c_x & c_y & c_z \end{bmatrix}$
$\mathbf{a}^T$	$\begin{bmatrix} \frac{EA}{l} & -\frac{EA}{l} \\ -\frac{EA}{l} & \frac{EA}{l} \end{bmatrix}$	
$\begin{bmatrix} c_x & 0 \\ c_y & 0 \\ c_z & 0 \\ 0 & c_x \\ 0 & c_y \\ 0 & c_z \end{bmatrix}$	$\begin{bmatrix} c_x\frac{EA}{l} & -c_x\frac{EA}{l} \\ c_y\frac{EA}{l} & -c_y\frac{EA}{l} \\ c_z\frac{EA}{l} & -c_z\frac{EA}{l} \\ -c_x\frac{EA}{l} & c_x\frac{EA}{l} \\ -c_y\frac{EA}{l} & c_y\frac{EA}{l} \\ -c_z\frac{EA}{l} & c_z\frac{EA}{l} \end{bmatrix}$	$\underbrace{\begin{bmatrix} c_x^2\frac{EA}{l} & c_yc_x\frac{EA}{l} & c_zc_x\frac{EA}{l} & -c_x^2\frac{EA}{l} & -c_yc_x\frac{EA}{l} & -c_zc_x\frac{EA}{l} \\ c_xc_y\frac{EA}{l} & c_y^2\frac{EA}{l} & c_zc_y\frac{EA}{l} & -c_xc_y\frac{EA}{l} & -c_y^2\frac{EA}{l} & -c_zc_y\frac{EA}{l} \\ c_xc_z\frac{EA}{l} & c_yc_z\frac{EA}{l} & c_z^2\frac{EA}{l} & -c_xc_z\frac{EA}{l} & -c_yc_z\frac{EA}{l} & -c_z^2\frac{EA}{l} \\ -c_x^2\frac{EA}{l} & -c_yc_x\frac{EA}{l} & -c_zc_x\frac{EA}{l} & c_x^2\frac{EA}{l} & c_yc_x\frac{EA}{l} & c_zc_x\frac{EA}{l} \\ -c_xc_y\frac{EA}{l} & -c_y^2\frac{EA}{l} & -c_zc_y\frac{EA}{l} & c_xc_y\frac{EA}{l} & c_y^2\frac{EA}{l} & c_zc_y\frac{EA}{l} \\ -c_zc_x\frac{EA}{l} & -c_yc_z\frac{EA}{l} & -c_z^2\frac{EA}{l} & c_xc_z\frac{EA}{l} & c_yc_z\frac{EA}{l} & c_z^2\frac{EA}{l} \end{bmatrix}}_{\mathbf{k}_G}$ (Feld S in STAN)

Elementnormalkraft N in Abhängigkeit der globalen Stabendverschiebungen **v**

$$\mathbf{S} = \mathbf{k}\,\mathbf{u} = \mathbf{k}\,\mathbf{a}\,\mathbf{v}$$

	$\mathbf{a}$	$\begin{bmatrix} c_x & c_y & c_z & 0 & 0 & 0 \\ 0 & 0 & 0 & c_x & c_y & c_z \end{bmatrix}$
$\mathbf{k}$		
$\begin{bmatrix} \frac{EA}{l} & -\frac{EA}{l} \\ -\frac{EA}{l} & \frac{EA}{l} \end{bmatrix}$		$\underbrace{\begin{bmatrix} \frac{EA}{l}c_x & \frac{EA}{l}c_y & \frac{EA}{l}c_z & -\frac{EA}{l}c_x & -\frac{EA}{l}c_y & -\frac{EA}{l}c_z \\ -\frac{EA}{l}c_x & -\frac{EA}{l}c_y & -\frac{EA}{l}c_z & \frac{EA}{l}c_x & \frac{EA}{l}c_y & \frac{EA}{l}c_z \end{bmatrix}}_{\text{Feld ST in STAN}}$

4.3 Direkte Steifigkeitsmethode und FE-Programmierung

Wiederholung FEM-Vorlesung; jedoch mit 3D-Fachwerkelement.

LOCATION-Matrix enthält Nummern N der Freiheitsgrade r_N der Struktur.

4.3.1 Knoteneingabe mit Randbedingungen (Subroutine INPUTJ)

IDENTITY-Feld ist eine Matrix mit NUMNP-Zeilen und NDOF-Spalten, wobei NUMNP („**NUM**ber of **N**odal **P**oints“) die Anzahl der Knoten der Struktur und NDOF („**N**umber of **D**egrees **O**f **F**reedom“) die Anzahl der Freiheitsgrade pro Knoten ist.

Die Elemente der IDENTITY_Matrix sind zuerst entweder „0“ oder „1“, wobei

1 = Lagerbedingung

0 = Freiheitsgrad existiert

Im Raum: 6 Knotenfreiheitsgrade: 3 Verschiebungen, 3 Drehungen

Beginnend bei 1 werden danach mit aufsteigenden natürlichen Zahlen die Freiheitsgrade **N** ersetzt. Alle fest vorgeschriebenen Knoten am Rand werden mit 0 besetzt.

Beispiel: Fachwerk

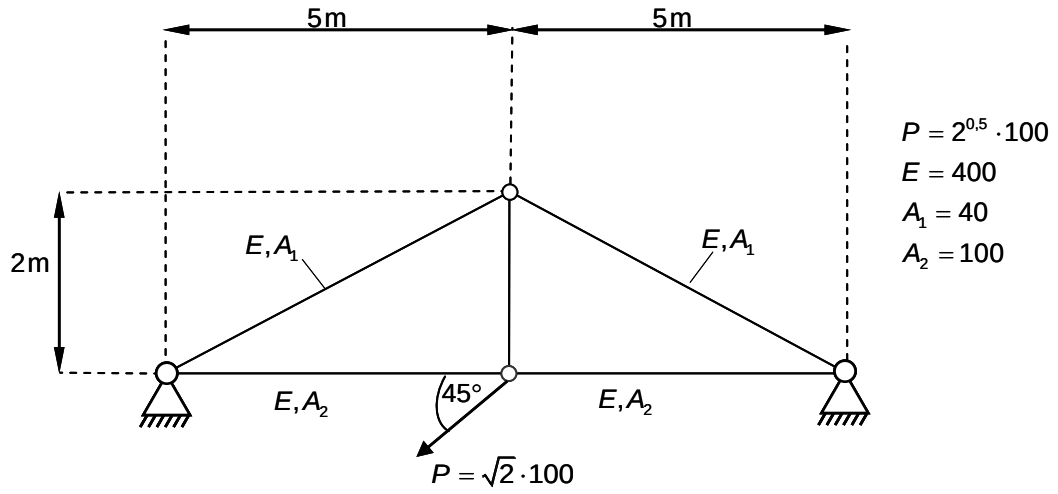


Abb. 4.3-1: Beispiel für Fachwerksysteme

Knoten- und Elementnummerierung:

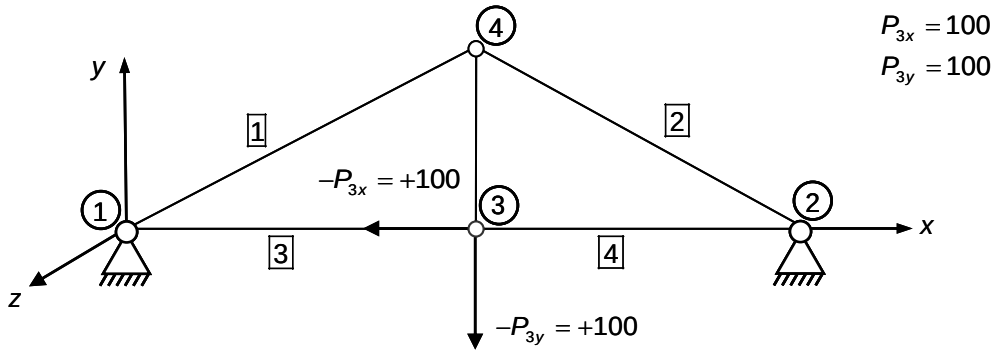


Abb. 4.3-2: Knoten- und Elementnummern

Knoten	Randbedingungen / Freiheitsgrade						Knotenkoordinaten		
Nr.	X	Y	Z	ZZ	YY	ZZ	X	Y	Z
1	1	1	1	1	1	1	0.000	0.000	0.000
2	1	1	1	1	1	1	10.000	0.000	0.000
3	0	0	1	1	1	1	5.000	0.000	0.000
4	0	0	1	1	1	1	5.000	2.000	0.000

ID-Feld ist eine Matrix mit NUMNP-Zeilen und NDOF-Spalten

Zuerst werden Randbedingungen eingegeben (0 oder 1):

0 → Freiheitsgrad existiert

1 → Randbedingung

Beginnend bei 1 werden mit aufsteigenden natürlichen Zahlen die Freiheitsgrade der Knoten nacheinander nummeriert.

Knoten	Freiheitsgradnummern					
Nr.	X	Y	Z	XX	YY	ZZ
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	1	2	0	0	0	0
4	3	4	0	0	0	0

SUBROUTINE INPUTJ (Knoteneingabe)

Knotenordinaten mit Knotenrandbedingungen einlesen:

```
READ  (NIN, 1000)  N, (ID (N,I), I=1,6),  X(N), Y(N), Z(N)
```

↑ „identity-array“ einlesen

Gleichungsnummern erzeugen:

```
          NEQ=0
          DO   60  N = 1, NUMNP
          DO   60  I = 1,6
          IF   (ID (N,I). GT. Ø )  GO  TO  58
          NEQ = NEQ +1
          ID (N,I) = NEQ
          GO TO 60
58        ID (N,I) = 0
60   CONTINUE
  ⋮
     DO  70   N =1, NUMNP
         WRITE (NOT, 2005)    N, (ID (N, I),  I = 1,6)
  ⋮
```

Anzahl der unbekannten Freiheitsgradnummern der Knoten, mit denen später die „Location“-Matrix besetzt wird.

- siehe unten -

4.3.2 Eingabe der Fachwerkelemente

Die Elementinformationen bestehen aus den Nummern des Anfangs- ⓘ und Endknotens ⓙ sowie der Elementnummer, dem Elastizitätsmodul für den Werkstoff und der Querschnittsfläche. Sie werden in der Subroutine TRUSS eingelesen.

```
READ (NIN, 1001)  M, I, J,  AREA, E
```

Aus den zuvor eingegebenen Knotenkoordinaten X(I), Y(I), Z(I) können die Länge, die Dehnsteifigkeit/Länge und die Richtungscosinus des Fachwerkelements bestimmt werden.

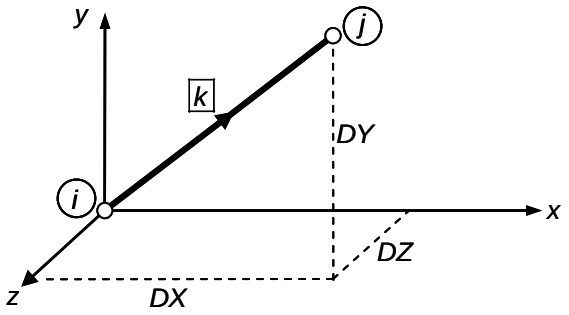
```
......DX = X(J) – X(I)
......DY = Y(J) – Y(I)
......DZ = Z(J) – Z(I)
......XL2 = DX * DX + DY * DY + DZ * DZ
......XL = SQRT (XL2)
......XX = E * AREA / XL
```

} - Relativer Ortsvektor

- Länge

- Dehnsteifigkeit

Die Richtungscosinus ergeben sich aus den Komponenten des relativen Ortsvektors und der Elementlänge



$$cx = \cos\left(\mathbf{e}_k, \mathbf{E}_x\right) = \frac{X_j - X_i}{l_{\boxed{k}}}$$

$$cy = \cos\left(\mathbf{e}_k, \mathbf{E}_y\right) = \frac{Y_j - Y_i}{l_{\boxed{k}}}$$

$$cz = \cos\left(\mathbf{e}_k, \mathbf{E}_z\right) = \frac{Z_j - Z_i}{l_{\boxed{k}}}$$

Abb. 4.3-3: Komponenten des relativen Ortsvektors für das Fachwerkelement

In SUBROUTINE TRUSS:

```
ST(1,1)=DX/XL        (cx)
ST(1,2)=DY/XL        (cy)
ST(1,3)=DZ/XL        (cz)
```

Siehe SUBROUTINE TRUSS für die Elementeingabe im Programm STAN

4.3.3 Bildung der Steifigkeitsmatrix (SUBROUTINE TRUSS im FE-Programm STAN)

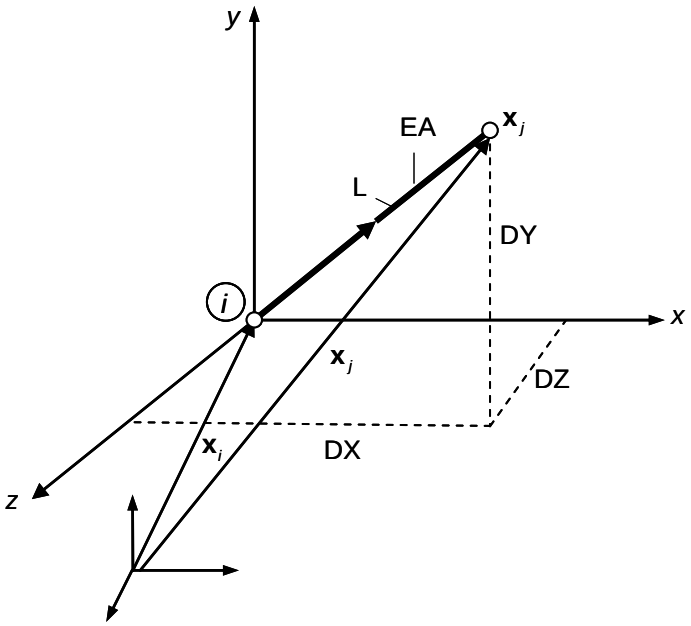


Abb. 4.3-4: Ortsvektor der Elementknoten

Ortsvektoren der Knoten (i) und (j) : $\mathbf{x}_i$, $\mathbf{x}_j$

Relativer Ortsvektor:

$$\Delta \mathbf{x} = \mathbf{x}_j - \mathbf{x}_i$$

Endknoten — Anfangsknoten

$$\Delta \mathbf{x} = \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} - \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} \begin{matrix} DX \\ DY \\ DZ \end{matrix}$$

in STAN

Stablänge: $l = \sqrt{\Delta X^2 + \Delta Y^2 + \Delta Z^2}$

Die Richtungscosinus (Cosinus des Richtungswinkels) ergeben sich aus dem Skalarprodukt des Basisvektors des lokalen Koordinatensystems in Richtung der Stabachse mit den Einheitsbasen des globalen Koordinatensystems

$$\cos\left(\mathbf{e}_i, \mathbf{E}_x\right) = \mathbf{e}_i \cdot \mathbf{E}_x = \frac{\Delta X}{l}$$

$$\cos\left(\mathbf{e}_i, \mathbf{E}_y\right) = \mathbf{e}_i \cdot \mathbf{E}_y = \frac{\Delta Y}{l}$$

$$\cos\left(\mathbf{e}_i, \mathbf{E}_z\right) = \mathbf{e}_i \cdot \mathbf{E}_z = \frac{\Delta Z}{l}$$

Steifigkeitsmatrix :

$$\mathbf{k}_G = \begin{bmatrix} cx^2\frac{EA}{l} & cycx\frac{EA}{l} & czcx\frac{EA}{l} & -cx^2\frac{EA}{l} & -cycx\frac{EA}{l} & -czcx\frac{EA}{l} \\ cxcy\frac{EA}{l} & cy^2\frac{EA}{l} & czcy\frac{EA}{l} & & & \\ cxcz\frac{EA}{l} & cxcy\frac{EA}{l} & cz^2\frac{EA}{l} & & & \\ -cx^2\frac{EA}{l} & & & cx^2\frac{EA}{l} & & \\ -cxcy\frac{EA}{l} & & & & cy^2\frac{EA}{l} & \\ -cxcz\frac{EA}{l} & & & & & cz^2cy^2\frac{EA}{l} \end{bmatrix}$$

Programmierung in SUBROUTINE TRUSS:

```
      ⋮

      DO 500 MM=1,NUME
        READ (NIN,1001)  M,I,J,AREA,E
        WRITE (NOT,2001) M,I,J,AREA,E
        DX=X(I)-X(J)
        DY=Y(I)-Y(J)
        DZ=Z(I)-Z(J)
        XL2=DX*DX+DY*DY+DZ*DZ
        XL=SQRT(XL2)
        XX=E*AREA/XL
        ST(1,1)=DX/XL
        ST(1,2)=DY/XL
        ST(1,3)=DZ/XL
        ST(1,4)=-ST(1,1)
        ST(1,5)=-ST(1,2)
        ST(1,6)=-ST(1,3)
C
        DO 300 L=1,6
        YY=ST (1,L)*XX
          DO 250 K=L,6
            S(K,L)=ST(1,K)*YY
 250        S(L,K)=S(K,L)
          ST(1,L)=YY
 300      ST(2,L)=YY/AREA
```

4.3.4 Zuordnung: Elementfreiheitsgrade (global) ↔ Strukturfreiheitsgrade

Aus dem IDENTITY-Feld ID(NUMNP,NDOF) werden die Zuordnungsvektoren $LM_{\boxed{i}}(ND)$ zwischen den Elementfreiheitsgraden $v_1,...,v_{ND}$ und den Knotenfreiheitsgraden $r_1,....,r_{NEQ}$ gebildet. Die Zuordnungsvektoren können in der LOCATION-Matrix LM (ND, NEL) zusammengefasst werden, wobei ND („**N**umber of **D**egrees of freedom“) die Anzahl der Elementfreiheitsgrade bedeutet und NEL („**N**umber of **EL**ements“) die Zahl der Elemente ist.

Beispiel: Fachwerk mit räumlichen Stabelementen in der X-Y-Ebene

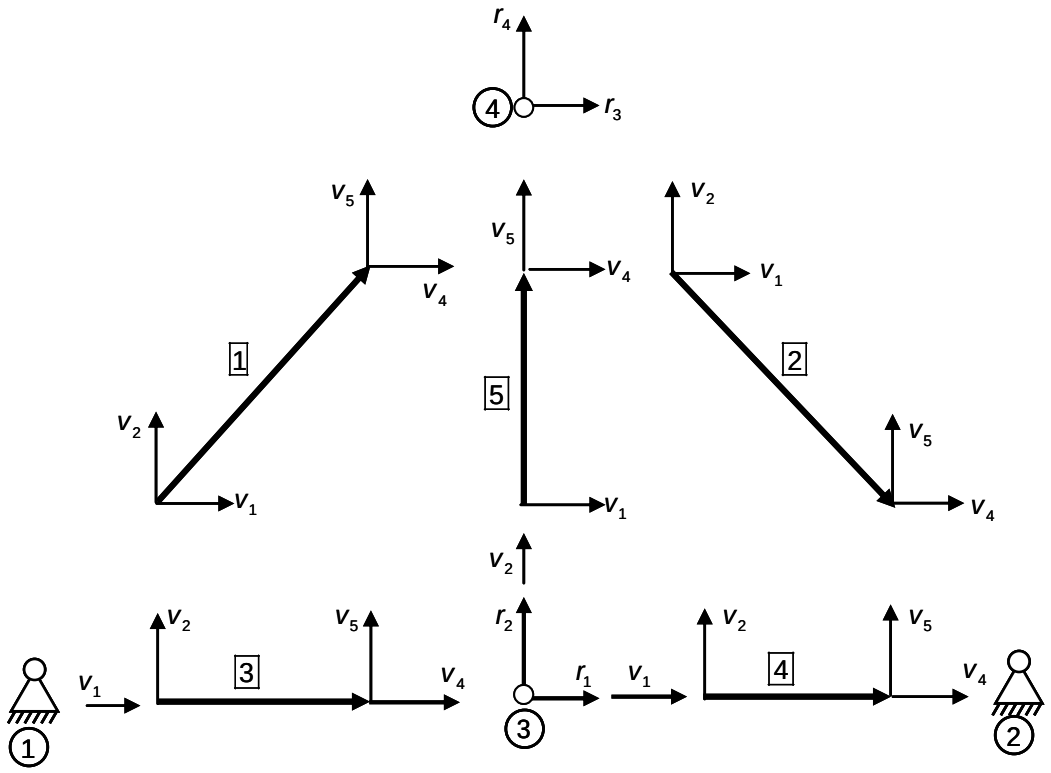


Abb. 4.3-5: Elementfreiheitsgrade

Zuordnungsvektor (≙ Spalte der Location Matrix) des Fachwerks $\boxed{k}$ mit Anfangsknoten I und Endknoten J:

$$LM_{\boxed{k}} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}_{\boxed{k}} = \begin{bmatrix} ID(I,1) \\ ID(I,2) \\ ID(I,3) \\ ID(J,1) \\ ID(J,2) \\ ID(J,3) \end{bmatrix}_{\boxed{k}}$$

Siehe: SUBROUTINE TRUSS

```
DO 500 MM=1,NUME
  READ (NIN,1001) M,I,J,AREA,E
  ...
  ...
  ...
 LM(1)=ID(I,1)
 LM(2)=ID(I,2)
 LM(3)=ID(I,3)
 LM(4)=ID(J,1)
 LM(5)=ID(J,2)
 LM(6)=ID(J,3)
  CALL CALBAN(NS,ND,LM,S,ST,MBAND)
  WRITE (NT2,*) (LM(I),I=1,ND),((ST(I,J),J=1,ND),I=1,NS)
```

```
500   CONTINUE
```

Beispiel:

Element $\boxed{1}$ des Fachwerks; Element $\boxed{2}$ Element $\boxed{3}$ Element $\boxed{4}$

$$\mathbf{LM}_{\boxed{1}}=\begin{bmatrix}0\\0\\0\\3\\4\\0\end{bmatrix} \quad \mathbf{LM}_{\boxed{2}}=\begin{bmatrix}3\\4\\0\\0\\0\\0\end{bmatrix} \quad \mathbf{LM}_{\boxed{3}}=\begin{bmatrix}0\\0\\0\\1\\2\\0\end{bmatrix} \quad \mathbf{LM}_{\boxed{4}}=\begin{bmatrix}1\\2\\0\\0\\0\\0\end{bmatrix}$$

4.3.5 Direkte Steifigkeitsoperationen

Anhand des Zuordnungsvektors $\mathrm{LM}_{\boxed{k}}$ für das Element $\boxed{k}$ kann dessen Elementsteifigkeit $\mathbf{k}_{Gij}$ auf die Struktursteifigkeit $\mathbf{K}_{MN}$ aufaddiert werden.

r_N r_M

Elementsteifigkeit

$$\mathrm{LM}^{T}_{\boxed{k}}=\begin{bmatrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{bmatrix}$$

$$\mathbf{k}_G=\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \\ & & k_{33} & k_{34} & k_{35} & k_{36} \\ & & & k_{44} & k_{45} & k_{46} \\ & \text{sym} & & & k_{55} & k_{56} \\ & & & & & k_{66} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} \begin{matrix} \\ \leftarrow r_N \\ \\ = \mathrm{LM}_{\boxed{k}} \\ \\ \leftarrow r_M \\ \\ \end{matrix}$$

Struktursteifigkeit

$$\mathbf{K}=\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ & K_{22} & K_{23} & K_{24} \\ \text{sym.} & & K_{33} & K_{34} \\ & & & K_{44} \end{bmatrix} \begin{matrix} \\ \\ \\ \leftarrow r_N \end{matrix}$$

(r_M ↓)

Zusammenbau der Elementsteifigkeit zur Struktursteifigekeitsmatrix:

$$\mathbf{K}=\bigcup_{k=1}^{nel}\mathbf{k}_G$$

Vereinigung über alle Elemente

Vorgehensweise

1. Berechne Elementsteifigkeiten $\mathbf{k}_G$ zwischen globalen Elementverschiebungen **v** und globalen Elementkräften $\mathbf{S}_G$
2. Bestimme die Gleichungsnummern derjenigen Strukturgleichungen, die vom vorliegenden Element betroffen sind.
3. Addition der Elementsteifigkeitsterme auf die entsprechenden Strukturgleichungen.

FORTRAN-Befehle für die direkten Steifigkeitsoperationen

- in SUBROUTINE ADDSTF

```
      SUBROUTINE ADDSTF(A,NEQ,MBAND,NUMTEL)
      DIMENSION A(NEQ,MBAND)
      COMMON LM(24),S(24,24)
      ...
      ...
      ...
      DO 300 I=1,ND
        N=LM(I)
        IF (N.EQ.0) GO TO 300
        DO 200 J=1,ND
          M=LM(J)-N+1
          IF(M.GT.0) THEN
               A(N,M)=A(N,M)+S(I,J)
          ENDIF
  200   CONTINUE
  300 CONTINUE
```

Strukturgleichungen **K** in Bandform Matrix der Zuordnungsvektoren und Elementsteifigkeit

(ND=6) Zahl der Freiheitsgrade pro Knoten

4.3.6 Bildung des Lastvektors aus den Knotenlasten

Die Knotenlasten auf dem Feld R(6) in der SUBROUTINE FORMLD werden in Form der kartesischen Komponenten P_{ix} und P_{iy} für jeden Lastfall L an den Knoten N eingelesen – siehe SUBROUTINE FORMLD – und danach im Strukturlastvektor D(NEQ, NUMLC) NEQ (**N**umber of **EQ**uations), NUMLC (**NUM**ber of **L**oad **C**ases)) für die rechten Seiten der einzelnen Lastfälle (mit direkter Steifigkeitsmethode) abgespeichert.

```
      SUBROUTINE FORMLD(D,ID,NEQ,NUMNP,NUMLC)
      DIMENSION D(NEQ,NUMLC),ID(NUMNP,6),R(6)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
c-----READ LOADS AND STORE IN LOAD MATRIX
      REWIND NT3
      Write (NOT,2001)
      DO 100 M=1,NEQ
        DO 100 L=1,NUMLC
  100     D(M,L)=0.0
      READ (NT3,*) ((ID(N,I),N=1,NUMNP),I=1,6)
C
  150 READ (NIN,1000) N,L,R
      IF(N.EQ.0) GO TO 300
      WRITE (NOT,2000) N,L,R
      DO 200 I=1,6
```

```
        II=ID(N,I)
        IF(II.EQ.0) GO TO 200
        D(II,L)=R(I)
  200   CONTINUE
      GO TO 150
C
  300 RETURN
 1000 FORMAT (2I5,6F10.0)
 2000 FORMAT (2I6,6F10.2)
 2001 FORMAT (/12H NODE  LOAD ,9X,1HX,9X,1HY,9X,1HZ,8X,2HXX,8X,
     1 2HYY,8X,2HZZ)
      END
```

Auf dem Feld **D**(NEQ, NUMLC) sind für jeden Lastfall die rechten Seiten **P** des Systems der Variationsgleichungen

$$\mathbf{Kr} = \mathbf{P}$$

gespeichert.

4.3.7 Rückrechnung zu den Schnittgrößen

Berechnung der Normalkraft (Stabkraft) der Fachwerkelemente aus den zuvor errechneten Knotenverschiebungen **v** (-siehe Kapitel 3.2)

$$\mathbf{s} = \mathbf{kav}$$

Im Hauptprogramm:

```
PROGRAMM STAN
⋮
CALL ELFOR
```

```
└→ SUBROUTINE   ELFOR
   ⋮
   CALL TRUSSF (A(NN))
   └→ SUBROUTINE   TRUSSF (D)
      ⋮
      CALL  CALSTR (2,6, LM, ST, SIG,D)
      └→ SUBROUTINE   CALSTR
```

In Subroutine CALSTR: Ermittlung der Knotenverschiebungsnummern und damit der Verschiebungsfreiheitsgrade für das betrachtete Element zur Berechnung der Stabkräfte, d.h. Normalkraft im Fachwerkstab.

$$\mathbf{ka} = \begin{bmatrix} \frac{EA}{l}cx & \frac{EA}{L}cy & \frac{EA}{l}cz & -\frac{EA}{l}cx & -\frac{EA}{l}cy & -\frac{EA}{l}cz \\ \text{.....nicht abgespeichert....} \end{bmatrix}$$

Die Matrix ist im Feld ST in SUBROUTINE CALSTR abgespeichert und dient zur Berechnung der Stabkraft (mit der ersten Zeile) und der Normalspannung (mit der zweiten Zeile der Matrix) des Fachwerkelements.

```
      SUBROUTINE CALSTR (NS,ND,LM,ST,SIG,D)
      DIMENSION LM(ND),ST(NS,ND),SIG(NS),D(1)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
      READ (NT2,*) (LM(I),I=1,ND),((ST(I,J),J=1,ND),I=1,NS)
      DO 600 I=1,NS
      SIG(I)=0
      DO 500 K=1,ND
      KK=LM(K)
      IF(KK.EQ.0) GO TO 500
      SIG(I)=SIG(I)+ST(I,K)*D(KK)
  500 CONTINUE
  600 CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE ELFOR(NUMLC)
      COMMON A(1)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C-----EVALUATION OF ELEMENT FORCES
      MM=N1+NEQ-1
      REWIND NT1
C
      DO 200 L=1,NUMLC
        WRITE (NOT,2001) L
        NN=N1+NEQ*(L-1)
        REWIND NT2
        DO 200 M=1,NELTYP
          READ (NT2,*) NPAR
          MTYPE=NPAR(1)
      GO TO (1,2,3),MTYPE
    1 CALL TRUSSF(A(NN))
      GO TO 200
c    2 CALL FRAMEF(A(NN))
    2 CONTINUE
    3 CONTINUE
  200 CONTINUE
C
      RETURN
 2001 FORMAT (/,30H MEMBER FORCES LOAD CONDITION ,I3)
      END
```

4.4 FE - Programm STAN

Das Programm STAN ist ein einfacher FORTRAN-Code für die Berechnung von Strukturen mittels der Methode der finiten Elemente. Hier liegt nur das Fachwerkelement im Code vor. STAN kann einfach um weitere Elemente aufgebaut werden. So wie hier abgedruckt, lassen sich nur räumliche Fachwerke berechnen. In der Veranstaltung wird ein Balkenelement entwickelt und in STAN implementiert, so dass ebene Rahmen mit drei Freiheitsgraden pro Knoten berechnet werden können.

4.4.1 Allgemeines zum FE - Programm STAN (STructural ANalysis)

STAN

Structural Analysis Program

By

Edward L. Wilson

University of California at Berkeley

SUMMARY

STAN is a small, simple, general purpose program for static load analysis of three-dimensional structural systems. Its major purpose is to illustrate the internal organization of computer programs for structural analysis. Since its major purpose is educational, efficient numerical techniques in many areas of the program have been omitted in order to make the program readable and simple to understand. Also, practical options such as large capacity, joint and member generation, member loads and internal member release have not been included. The addition of new numerical methods and other practical options are best accomplished by individual student projects.

INPUT DATA

The first step in the analysis of a structure by a computer program is to number all joints and members in the system. The next step is to identify the displacement degrees of freedom at the joints. In addition each joint and each member requires one identification line of information. The positive definition of joint displacements and loads are shown bellow:

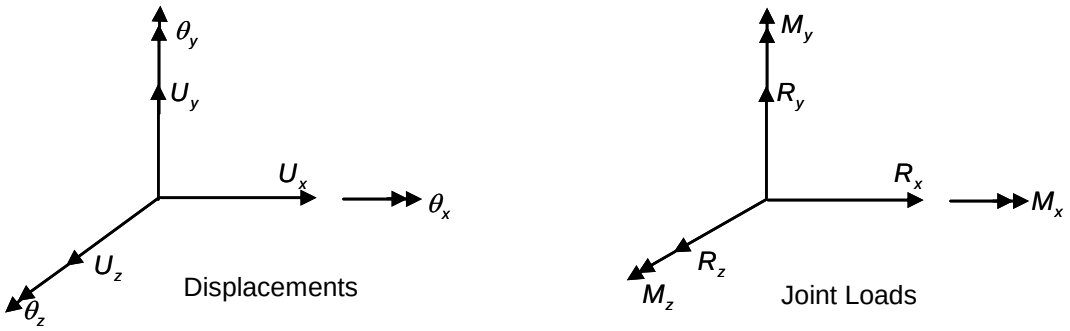


Abb. 4.4-1: Definition der Knotenverschiebungen und -schnittgrößen

4.4.2 Formatierte Dateneingabe in STAN

The input data is prepared and is placed on a file named "INPUT". The following sequence of input data provides a numerical definition of the structure:

HEADING LINE (80A1)

This line is used to identify the structure to be solved.

MASTER CONTROL LINE (3I5)

1 – 5 Total number of joints in the structural system.

6 – 10 Number of different types of elements.

11 – 15 Number of different load conditions.

JOINT DATA (7I5,3F10.0)

1 – 5 Joint number to be selected by the user.

6 – 10 U_x

11 – 15 U_y

16 – 20 U_z

21 – 25 θ_x

26 – 30 θ_y

31 – 35 θ_z

Displacement Boundary Condition Code

0 – displacement exists

1 – displacement is zero

or does not exist

36 – 45 x-Ordinate

46 – 55 y-Ordinate

56 – 65 z-Ordinate

ELEMENT DATA

One set of data must be supplied for each different element type according to the format given below:

Three-Dimensional Truss Members – TYPE 1

First Line – Truss member control line (2I5)

1 – 5 The number 1

6 – 10 Number of Truss members

Truss Member Lines (3I5,2F10.0) – One line for each member

1 – 5 Member identification number

6 – 10 Joint number I

11 – 15 Joint number J

16 – 30 Area of truss member, A

31 – 45 Modulus of elasticity, E

Two-Dimensional Frame Members – TYPE 2 Members

Type 2 indicates bending members in the X-Y Plane

First Line

1 – 5 The number 2

6 – 10 Number of 2-D Members

Frame Member Lines (3I5,5F10.0) – One line for each member

1 – 5 Member identification number

6 – 10 Joint number I

11 – 15 Joint number J

16 – 25 Axial Area of member

26 – 35 Moment of Inertia

46 – 55 Modulus of Elasticity

56 – 65 Shear Area

66 – 75 Shear Modulus

JOINT LOADS

One line must be supplied for each loaded joint and for each load condition. Data must be entered in load condition order.

1 – 5 Joint number

6 – 10 Load condition number

11 – 20 R_x

21 – 30 R_y

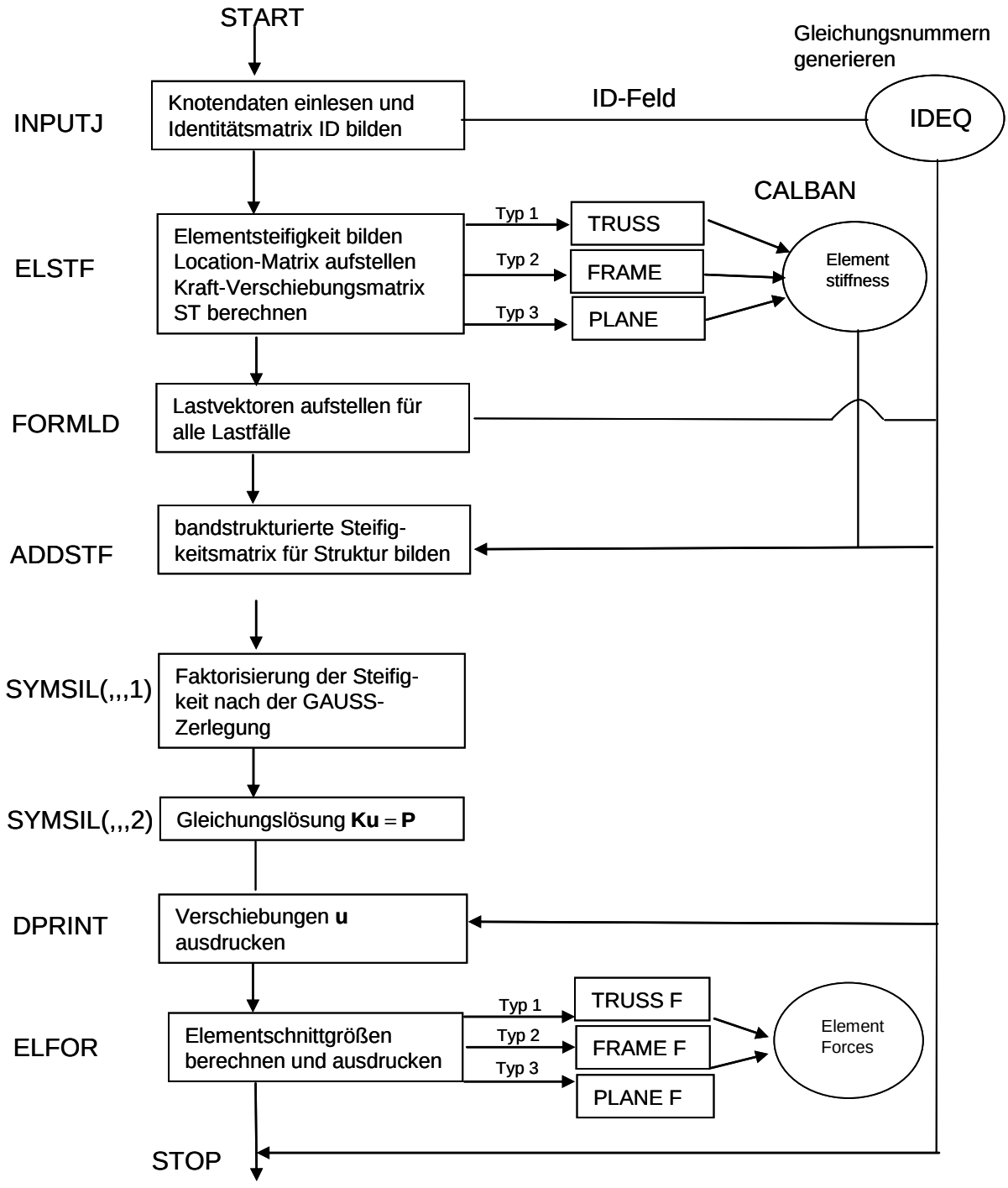
31 – 40 R_z

41 – 50 M_x

51 – 60 M_y

61 – 70 M_z

4.4.3 Organisation des Programms – Programmablauf



4.4.4 Beispiele zur Dateneingabe

Jeweils ein Beispiel für die Berechnung eines Fachwerks und eines Rahmens mit dem Programm STAN werden nachfolgend angegeben.

Beispiel 1: Unterspannter Träger

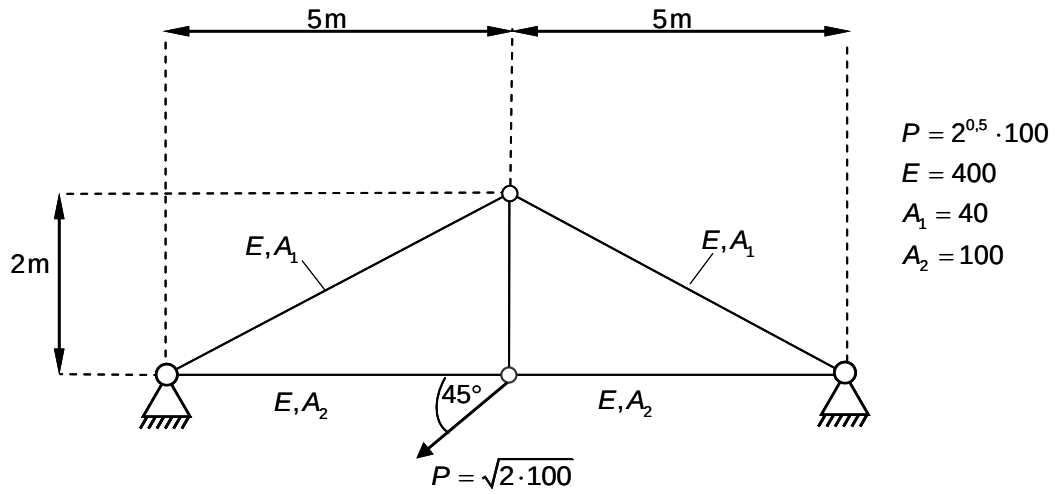


Abb. 4.4-2: Systemskizze mit Randbedingungen und Lasten

Datensatz für Eingabe in STAN

```
Unterspannter Traeger
4     2     1
1     1     1     1     1     1     1     0           0           0
2     1     1     1     1     1     1     10          0           0
3     0     0     1     1     1     1     5           0           0
4     0     0     1     1     1     1     5           2           0
1     2
1     1     4      40              400
2     4     2      40              400
1     3
1     1     3      100             400
2     3     2      100             400
3     3     4      100             400
3     1     -100         -100       0          0          0          0
0     0      0           0          0          0          0          0
```

Ausgabedaten von STAN zum unterspannten Träger

```
unterspannter Träger

 Number of Joints                          =   4

 Number of different element types         =   2

 Number of load conditions                 =   1

 NODE   BOUNDARY CONDITION CODES           NODAL POINT COORDINATES

 NUMBER  X     Y     Z    XX    YY    ZZ          X            Y            Z

    1    1     1     1     1     1     1        .000         .000         .000

    2    1     1     1     1     1     1      10.000         .000         .000

    3    0     0     1     1     1     1       5.000         .000         .000

    4    0     0     1     1     1     1       5.000        2.000         .000
```

EQUATION NUMBERS

N	X	Y	Z	XX	YY	ZZ
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	1	2	0	0	0	0
4	3	4	0	0	0	0

TRUSS MEMBERS

NUMBER	I	J	AREA	E
1	1	4	40.000	.400000E+03
2	4	2	40.000	.400000E+03

TRUSS MEMBERS

NUMBER	I	J	AREA	E
1	1	3	100.000	.400000E+03
2	3	2	100.000	.400000E+03
3	3	4	100.000	.400000E+03

NODE	LOAD	X	Y	Z	XX	YY	ZZ
3	1	-100.00	-100.00	.00	.00	.00	.00

FORMATION OF STIFFNESS MATRIX

START OF SOLUTION OF EQUATIONS

DISPLACEMENTS FOR LOAD CONDITION 1

NODE	X	Y	Z	XX	YY	ZZ
1	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00
2	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00
3	-.62500E-02	-.12701E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00
4	.00000E+00	-.12201E+00	.00000E+00	.00000E+00	.00000E+00	.00000E+00

MEMBER FORCES LOAD CONDITION 1

```
 MEMBER          FORCE          STRESS
     1   -.134629E+03   -.336573E+01
     2   -.134629E+03   -.336573E+01
 MEMBER          FORCE          STRESS
     1   -.500000E+02   -.500000E+00
     2    .500000E+02    .500000E+00
     3    .100000E+03    .999999E+00
```

Beispiel 2: Rahmen

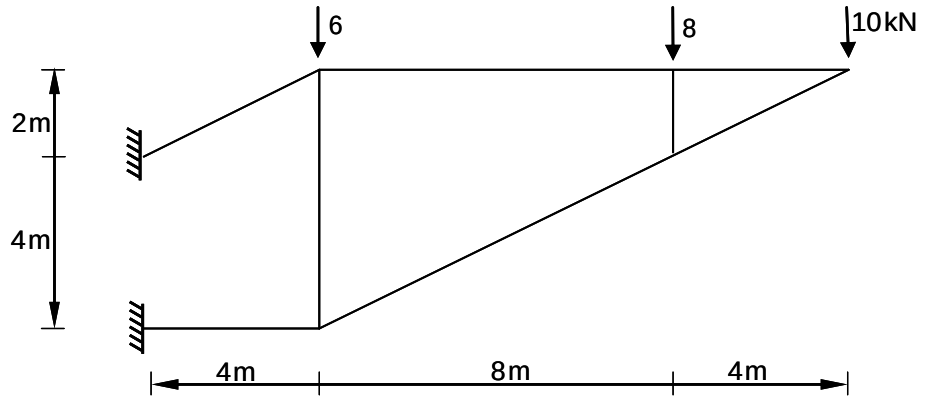


Abb. 4.4-3: Systemskizze des Rahmens

Datensatz für Eingabe in STAN

```
Rahmen
7    1    1
1    1    1    1    1    1    1    0          4000       0
2    0    0    1    1    1    0    4000       6000       0
3    0    0    1    1    1    0    12000      6000       0
4    0    0    1    1    1    0    16000      6000       0
5    0    0    1    1    1    0    12000      4000       0
6    0    0    1    1    1    0    4000       0          0
7    1    1    1    1    1    1    0          0          0
2    8
1    1    2     4000        200000    100000    0         0
2    2    3     4000        200000    100000    0         0
3    3    4     4000        200000    100000    0         0
4    2    6     4000        200000    100000    0         0
5    3    5     4000        200000    100000    0         0
6    7    6     4000        200000    100000    0         0
7    6    5     4000        200000    100000    0         0
8    5    4     4000        200000    100000    0         0
2    1    0          6000      0         0         0          0
3    1    0          8000      0         0         0          0
4    1    0          10000     0         0         0          0
0
```

Ausgabedaten von STAN zum Rahmen

```
Rahmen
```

Number of Joints = 7

Number of different element types = 1

Number of load conditions = 1

NODE	BOUNDARY CONDITION CODES						NODAL POINT COORDINATES		
NUMBER	X	Y	Z	XX	YY	ZZ	X	Y	Z
1	1	1	1	1	1	1	.000	4000.000	.000
2	0	0	1	1	1	0	4000.000	6000.000	.000
3	0	0	1	1	1	0	12000.000	6000.000	.000
4	0	0	1	1	1	0	16000.000	6000.000	.000
5	0	0	1	1	1	0	12000.000	4000.000	.000
6	0	0	1	1	1	0	4000.000	.000	.000
7	1	1	1	1	1	1	.000	.000	.000

EQUATION NUMBERS

N	X	Y	Z	XX	YY	ZZ
1	0	0	0	0	0	0
2	1	2	0	0	0	3
3	4	5	0	0	0	6
4	7	8	0	0	0	9
5	10	11	0	0	0	12
6	13	14	0	0	0	15
7	0	0	0	0	0	0

FRAME MEMBERS

NUMBER	I	J	AREA	XM	E	AS	G
1	1	2	4000.000	100000.000	200000.000	.000	.000
2	2	3	4000.000	100000.000	200000.000	.000	.000
3	3	4	4000.000	100000.000	200000.000	.000	.000
4	2	6	4000.000	100000.000	200000.000	.000	.000
5	3	5	4000.000	100000.000	200000.000	.000	.000
6	7	6	4000.000	100000.000	200000.000	.000	.000
7	6	5	4000.000	100000.000	200000.000	.000	.000
8	5	4	4000.000	100000.000	200000.000	.000	.000

NODE	LOAD	X	Y	Z	XX	YY	ZZ
2	1	.00	-6000.00	.00	.00	.00	.00
3	1	.00	-8000.00	.00	.00	.00	.00
4	1	.00	-10000.00	.00	.00	.00	.00

```
 FORMATION OF STIFFNESS MATRIX


 START OF SOLUTION OF EQUATIONS

 DISPLACEMENTS FOR LOAD CONDITION   1
 NODE          X           Y           Z          XX          YY          ZZ
    1  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
    2  .56658E+04 -.11331E+05  .00000E+00  .00000E+00  .00000E+00 -.24575E+01
    3  .56662E+04 -.23659E+05  .00000E+00  .00000E+00  .00000E+00  .85359E-01
    4  .56663E+04 -.22665E+05  .00000E+00  .00000E+00  .00000E+00  .32250E+00
    5  .61632E+04 -.23659E+05  .00000E+00  .00000E+00  .00000E+00  .11683E+00
    6 -.20862E+00 -.11331E+05  .00000E+00  .00000E+00  .00000E+00 -.25707E+01
    7  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00

 MEMBER FORCES LOAD CONDITION   1


 MEMBER          MI          MJ          P
     1    .540323E+08    .320517E+08    .367060E+05
     2   -.103379E+07    .116805E+08    .310305E+05
     3   -.251983E+07   -.148439E+06    .221490E+05
     4   -.310179E+08   -.317727E+08   -.656027E+04
     5   -.916069E+07   -.853122E+07   -.666525E+04
     6    .592767E+08    .335695E+08   -.417241E+05
     7   -.179678E+07    .102223E+08   -.352403E+05
     8   -.169110E+07    .148431E+06   -.246501E+05
```

4.4.5 Quellcode des FE – Programms STAN

```
  PROGRAM STAN
c     ------------------------------------------------------------------
c     THIS PROGRAM IS A SIMPLIFIED VERSION OF THE GENERAL STRUCTURAL
c     ANALYSIS PROGRAM SAP BY E. L. WILSON --- TO BE USED FOR EDUCAIIONAL
C     Purposes                                                    22.06.99
c     ------------------------------------------------------------------
c     Variablendeklaration:
c       NT1,NT2,NT3,NIN,NOT    Kanalnummer
c       NUMNP                 Anzahl der Knotenpunkte des Systems
c       MBAND                  Halbe Bandbreite
c       NELTYP                Anzahl der verschiedenen Elemente
c       NUMLC                 Anzahl der Lastfaelle
```

```
c       N1,N2,N3,N4,N5        Feldadressen fuer Belegung des blank common
c       MTOT                  maximale Feldlaenge des blank common
c       NEQ                   Anzahl der Freiheitsgrade
c       NPAR                  Anzahl der Materialparameter
c
      COMMON A(2000)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
c
      MBAND=0
      MTOT=2000
      NT1=16
      NT2=17
      NT3=18
      NIN=19
      NOT=13
c
      OPEN(NIN,file='INPUT')
      OPEN(NT1,file='ELSTIFF')
      OPEN(NT2,file='ELFORCES ')
      OPEN(NT3,file='IDEQ     ')
      OPEN(NOT,file='ausgabe     ')
      N1=601
      NUMTEL=0
c-----READ GENERAL CONTROL DATA FOR STRUCTURE
      READ (NIN,1000) (A(I),I=1,12),NUMNP,NELTYP,NUMLC
      WRITE(NOT,2000) (A(I),I=1,12),NUMNP,NELTYP,NUMLC
c-----READ JOINT DATA AND WRITE ID ARRAY ON TAPE NT3
      N2=N1+NUMNP
      N3=N2+NUMNP
      N4=N3+NUMNP
      N5=N4+6*NUMNP-1
      IF(N5.GT.MTOT) CALL ERROR(N5-MTOT)
      CALL INPUTJ(A(N1),A(N2),A(N3),A(N4),NUMNP,NEQ)
c-----FORM MEMBER MATRICES-STIFFNESS ON FILE NT1 AND STRESS ON FILE NT2
      REWIND NT1
      REWIND NT2
      DO 100 M=1,NELTYP
      READ (NIN,1001) NPAR
      WRITE (NT2,*) NPAR
c
      CALL ELSTF
  100 NUMTEL=NUMTEL+NPAR(2)
c-----READ LOAD CONDITIONS
      N2=N1+NEQ*NUMLC
      CALL FORMLD(A(N1),A(N2),NEQ,NUMNP,NUMLC)
c-----FORM STRUCTURE STIFFNESS MATRIX IN CORE
      N3=N2+NEQ*MBAND
      IF(N3.GT.MTOT) CALL ERROR(N3-MTOT)
      CALL ADDSTF(A(N2),NEQ,MBAND,NUMTEL)
c-----TRIANGULARISE STIFFNESS MATRIX
        CALL SYMSOL(A(N2),A(N1),NEQ,MBAND,1)
C-----CALCULATE DISPLACEMENTS
        DO 150 L=1,NUMLC
        NN=N1+NEQ*(L-1)
        CALL SYMSOL(A(N2),A(NN),NEQ,MBAND,2)
  150 CONTINUE
C-----PRINT DISPLACEMENTS AND MEMPER FORCES
         CALL DPRINT(A(N1),A(N2),NEQ,NUMNP,NUMLC)
        CALL ELFOR(NUMLC)
```

```
      STOP
C
1000  FORMAT(12A4/3I5)
1001  FORMAT(14I5)
2000  FORMAT (1H ,12A4/
     1' Number of Joints                              =',I4/
     2' Number of different element types             =',I4/
     3' Number of load conditions                     =',I4)
      END
C-----------------------------------------------------------------------
      SUBROUTINE INPUTJ(X,Y,Z,ID,NUMNP,NEQ)
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,6)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C-----READ JOINT DATA
      REWIND NT3
      WRITE (NOT,2001)
      DO 50 M=1,NUMNP
        READ  (NIN,1000) N,(ID(N,I),I=1,6),X(N),Y(N),Z(N)
        WRITE (NOT,2000) N,(ID(N,I),I=1,6),X(N),Y(N),Z(N)
   50 CONTINUE
C-----REPLACE ID ARRAY WITH EQUATION NUMBERS
      NEQ=0
      DO 60 N=1,NUMNP
        DO 60 I=1,6
          IF(ID(N,I).GT.0) THEN
            ID(N,I)= 0
          ELSE
            NEQ=NEQ+1
            ID(N,I)=NEQ
          ENDIF

   60 CONTINUE
      WRITE (NOT,2004)
      DO 70 N=1,NUMNP
        WRITE (NOT,2005) N,(ID(N,I),I=1,6)
   70 CONTINUE
      WRITE (NT3,*) ((ID(N,I),N=1,NUMNP),I=1,6)
      REWIND NT3
C
      RETURN
 1000 FORMAT (7I5,3F10.0)
 2001 FORMAT (5H NODE,3X,24HBOUNDARY CONDITION CODES,11X,
     .23HNODAL POINT COORDINATES / 7H NUMBER,2X ,1HX,4X,1HY,4X,1HZ,3X,
     .2HXX,3X,2HYY,3X,2HZZ,12X,1HX,12X,1HY,12X,1HZ)
 2000 FORMAT (I5,6I5,3F13.3)
 2004 FORMAT (//17H EQUATION NUMBERS/
     1 35H    N    X    Y   Z   XX   YY    ZZ)
 2005 FORMAT (7I5)
      END
C-----------------------------------------------------------------------
      SUBROUTINE ERROR(N)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
      WRITE (NOT, 2000) N
 2000 FORMAT (// 20H STORACE EXCEEDED BY ,I6)
      STOP
      END
C-----------------------------------------------------------------------
      SUBROUTINE ELSTF
      COMMON A(1)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
```

```
      MTYPE=NPAR(1)
c-----CALL APPROPRIATE ELEMENT
      GO TO (1,2,3), MTYPE3
c       THREE DIMENSIONAL TRUSS ELEMENTS
    1 CALL TRUSS(A(N1),A(N2),A(N3),A(N4),NUMNP)
        GO TO 900
c       TWO-DIMENSIONAL FRAME IN X-Y PLANE I
c    2 CALL FRAME(A(N1),A(N2),A(N3),A(N4),NUMNP)
    2 CONTINUE
        GO TO 900
c       PLANE STRESS ELEMENTS
c     3 CALL PLANE
    3 CONTINUE
  900 RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE ADDSTF(A,NEQ,MBAND,NUMTEL)
      DIMENSION A(NEQ,MBAND)
      COMMON LM(24),S(24,24)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C-----ZERO STIFFNESS MATRIX
      WRITE (NOT,2000)
      DO 100 N=1,NEQ
        DO 100 M=1,MBAND
  100   A(N,M)=0.0
C     READ ELEMENT STIFFNESS-FROM FILE 1 AND ADD TO STRUCTRE STIFFNESS
      REWIND NT1
      DO 400 L=1,NUMTEL
      READ (NT1,*) ND,(LM(I),I=1,ND),((S(I,J),J=1,ND),I=1,ND)
C
      DO 300 I=1,ND
        N=LM(I)
        IF (N.EQ.0) GO TO 300
        DO 200 J=1,ND
          M=LM(J)-N+1
          IF(M.GT.0) THEN
               A(N,M)=A(N,M)+S(I,J)
          END IF
  200 CONTINUE
  300 CONTINUE
  400 CONTINUE
C
 2000 FORMAT (/' FORMATION OF STIFFNESS MATRIX'/)
      END
C-----------------------------------------------------------------
      SUBROUTINE CALBAN(NS,ND,LM,S,ST,MBAND)
      DIMENSION LM(ND),S(ND,ND),ST(NS,ND)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C-----CHECK FOR MAXIMUM BAND WIDTH
      DO 450 I=1,ND
        IF(LM(I).EQ.0) GO TO 450
        DO 440 J=I,ND
          IF(LM(J).EQ.0) GO TO 440
          MAX=IABS(LM(I)-LM(J))+1
          IF(MAX.GT.MBAND) MBAND=MAX
  440 CONTINUE
  450 CONTINUE
      WRITE (NT1,*) ND,(LM(I),I=1,ND),((S(I,J),J=1,ND),I=1,ND)
      RETURN
      END
```

```
C-----------------------------------------------------------------
      SUBROUTINE CALSTR (NS,ND,LM,ST,SIG,D)
      DIMENSION LM(ND),ST(NS,ND),SIG(NS),D(1)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
      READ (NT2,*) (LM(I),I=1,ND),((ST(I,J),J=1,ND),I=1,NS)
      DO 600 I=1,NS
      SIG(I)=0
      DO 500 K=1,ND
      KK=LM(K)
      IF(KK.EQ.0) GO TO 500
      SIG(I)=SIG(I)+ST(I,K)*D(KK)
  500 CONTINUE
  600 CONTINUE
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE ELFOR(NUMLC)
      COMMON A(1)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C-----EVALUATION OF ELEMENT FORCES
      MM=N1+NEQ-1
      REWIND NT1
C
      DO 200 L=1,NUMLC
        WRITE (NOT,2001) L
        NN=N1+NEQ*(L-1)
        REWIND NT2
        DO 200 M=1,NELTYP
          READ (NT2,*) NPAR
          MTYPE=NPAR(1)
      GO TO (1,2,3),MTYPE
    1 CALL TRUSSF(A(NN))
      GO TO 200
c    2 CALL FRAMEF(A(NN))
    2 CONTINUE
    3 CONTINUE
  200 CONTINUE
C
      RETURN
 2001 FORMAT (/,30H MEMBER FORCES LOAD CONDITION ,I3)
      END
C-----------------------------------------------------------------
      SUBROUTINE SYMSOL(A,B,NN,MM,KKK)
      DIMENSION A (NN,MM), B(NN)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C
      GO TO (1000,2000),KKK
C-----REDUCE-MATRIX
 1000 WRITE (NOT,3000)
      DO 280 N=1,NN
      IF(A(N,1).LE.0.0) WRITE (NOT,3001) N,A(N,1)
      DO 260 L=2,MM
      C=A(N,L)/A(N,1)
      I=N+L-1
      IF(NN-I) 260,240,240
  240 J=0
      DO 250 K=L,MM
      J=J+1
  250 A(I,J)=A(I,J)-C*A(N,K)
```

```
  260 A(N,L)=C
  280 CONTINUE
      GO TO 500
C-----REDUCE VECTOR
 2000 DO 290 N=1,NN
      DO 285 L=2,MM
      I=N+L-1
      IF(NN-I) 290,285,285
  285 B(I)=B(I)-A(N,L)*B(N)
  290 B(N)=B(N)/A(N,1)
C-----BACK SUBSTITUTION
      N=NN
  300 N=N-1
      IF(N) 350,500,350
  350 DO 400 K=2,MM
      L=N+K-1
      IF(NN-L) 400,370,370
  370 B(N) = B(N) - A(N,K)*B(L)
  400 CONTINUE
      GO TO 300
C
  500 RETURN
 3000 FORMAT (/' START OF SOLUTION OF EQUATIONS')
 3001 FORMAT (1X,' EQUATION # -',I5,3X,' DIAGONAL TERM = ',1E15.7/)
      END
C-----------------------------------------------------------------
      SUBROUTINE FORMLD(D,ID,NEQ,NUMNP,NUMLC)
      DIMENSION D(NEQ,NUMLC),ID(NUMNP,6),R(6)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
c-----READ LOADS AND STORE IN LOAD MATRIX
      REWIND NT3
      Write (NOT,2001)
      DO 100 M=1,NEQ
        DO 100 L=1,NUMLC
  100     D(M,L)=0.0
      READ (NT3,*) ((ID(N,I),N=1,NUMNP),I=1,6)
C
  150 READ (NIN,1000) N,L,R
      IF(N.EQ.0) GO TO 300
      WRITE (NOT,2000) N,L,R
      DO 200 I=1,6
        II=ID(N,I)
        IF(II.EQ.0) GO TO 200
        D(II,L)=R(I)
  200   CONTINUE
      GO TO 150
C
  300 RETURN
 1000 FORMAT (2I5,6F10.0)
 2000 FORMAT (2I6,6F10.2)
 2001 FORMAT (/12H NODE  LOAD ,9X,1HX,9X,1HY,9X,1HZ,8X,2HXX,8X,
     1 2HYY,8X,2HZZ)
      END
C-----------------------------------------------------------------
      SUBROUTINE DPRINT(D,ID,NEQ,NUMNP,NUMLC)
      DIMENSION D(NEQ,NUMLC),ID(NUMNP,6)
      COMMON DD(6)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C-----PRINT DISPLACEMENTS FOR ALL LOAD CONDITIONS
      REWIND NT3
```

```
      READ (NT3,*) ((ID(N,I),N=1,NUMNP),I=1,6)
C
      DO 200 L=1,NUMLC
        WRITE (NOT,2000) L
        DO 200 N=1,NUMNP
          DO 100 I=1,6
            DD(I)=0.0
            II=ID(N,I)
            IF(II.GT.0)        DD(I)=D(II,L)
  100       CONTINUE
  200     WRITE (NOT,2001) N,DD
C
      RETURN
 2000 FORMAT (/34H DISPLACEMENTS FOR LOAD CONDITION ,I3/5H NODE,11X,1HX
     1 ,11X,1HY,11x,1HZ,10X,2HXX,10X,2HYY,10X,2HZZ)
 2001 FORMAT (I5,6E12.5)
      END
C-----------------------------------------------------------------------
      SUBROUTINE TRUSS ( X, Y, Z, ID, NUMNP )
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,6)
      COMMON /ELPAR/ NPAR(14),NNNNN,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON S(6,6),ST(2,6),LM(6)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
      ND=6
      NS=2
      NUME=NPAR(2)
      WRITE (NOT,2000) NUME
      DO 500 MM=1,NUME
        READ (NIN,1001)  M,I,J,AREA,E
        WRITE (NOT,2001) M,I,J,AREA,E
        DX=X(J)-X(I)
        DY=Y(J)-Y(I)
        DZ=Z(J)-Z(I)
        XL2=DX*DX+DY*DY+DZ*DZ
        XL=SQRT(XL2)
        XX=E*AREA/XL
        ST(1,1)=DX/XL
        ST(1,2)=DY/XL
        ST(1,3)=DZ/XL
        ST(1,4)=-ST(1,1)
        ST(1,5)=-ST(1,2)
        ST(1,6)=-ST(1,3)
C
        DO 300 L=1,6
          YY=ST(1,L)*XX
          DO 250 K=L,6
            S(K,L)=ST(1,K)*YY
  250       S(L,K)=S(K,L)
          ST(1,L)=YY
  300     ST(2,L)=YY/AREA
        LM(1)=ID(I,1)
        LM(2)=ID(I,2)
        LM(3)=ID(I,3)
        LM(4)=ID(J,1)
        LM(5)=ID(J,2)
        LM(6)=ID(J,3)
        CALL CALBAN(NS,ND,LM,S,ST,MBAND)
        WRITE (NT2,*) (LM(I),I=1,ND),((ST(I,J),J=1,ND),I=1,NS)
  500   CONTINUE
      RETURN
```

```
 1001 FORMAT (3I5,2F15.0)
 2000 FORMAT (// I4, 13HTRUSS MEMBERS/
     1 ,7H NUMBER,6X,1HI,6X,1HJ,11X,4HAREA,14X,1HE)
 2001 FORMAT (3I7,F15.3,E15.6)
      END
C-----------------------------------------------------------------
      SUBROUTINE TRUSSF(D)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON LM(6),ST(2,6),SIG(2)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
  400 NUME=NPAR(2)
      WRITE (NOT,2000)
      DO 800 MM=1,NUME
        CALL CALSTR(2,6,LM,ST,SIG,D)
  800   WRITE (NOT,2001)MM,SIG(1),SIG(2)
      RETURN
 2000 FORMAT (//7H MEMBER,10X,5HFORCE,10X,6HSTRESS)
 2001 FORMAT (I6,2E15.6)
      END
c -----------------------------------------------------------------
      SUBROUTINE MPRINT(A,NR,NC)
C.... GENERAL MATRIX PRINT SUBROUTINE
      INTEGER NR,NC,J,JH,I,K
      REAL A(NR,NC)
C
      WRITE(7,2000) (I,I = 1,NC)
      DO 100 J = 1,NC,5
         JH = J + 4
         IF(JH.GT.NC) JH = NC
         DO 100 I = 1,NR
 100        WRITE(7,2001) I, (A(I,K), K = J,JH)
      RETURN
C.... FORMATS
 2000 FORMAT(/,'MATRIX ' /(8X,5I14))
 2001 FORMAT(I4,4X,5E14.7)
      END
```

4.4.6 Programmspezifisches in STAN

4.4.6.1 Berechnung der halben Bandbreite der Struktursteifigkeitsmatrix

Die maximale halbe Bandbreite MBAND ist die größte Differenz zwischen den Gleichungsnummern für zwei Freiheitsgrade in einem Element. Sie wird in der Subroutine CALBAN (CALculate BANdwidth) wie folgt mit einer DO-Schleife berechnet:

```
      ⋮
      DO 450 I = 1, ND                        (ND = 6  Knotenfreiheitsgrade)
        IF (LM (I).EQ. Ø)    GO  TO  450
        DO 440 J = I, ND
            IF (LM (J) . EQ. Ø)  GO TO 440
            MAX = IABS (LM(I) – LM(J)) +1
            IF (MAX.GT.MBAND)  MBAND = MAX
440     CONTINUE
450   CONTINUE
```

4.4.6.2 Dynamische Speicherplatzverwaltung

Bevor ein Feld in FORTRAN 77 angelegt wird, muss es fest dimensioniert werden; d.h. der Länge der Felder muss ein fester Zahlenwert zugewiesen werden, damit das Feld im „blank common“ vorgehalten werden kann.

Die Zahl der Knoten, Elemente, Lastfälle usw. sind jedoch eine problemabhängige Größen und bestimmten die Länge der Felder für die Speicherung der Knotenkoordinaten, Elementeigenschaften und Lastvektoren. Die erforderlichen Feldlängen werden aus den Kontrolldaten (Zahl der Knoten, Zahl der Elemente, Zahl der Materialien, Zahl der Lastfälle etc.) berechnet und daraus die Dimensionierung der Felder vorgenommen. Drei Möglichkeiten stehen zur Wahl – siehe Kapitel 2.5:

i) **Adressrechnung und Aufteilung des „blank common“** (halbdynamische Felder)

blank common: A

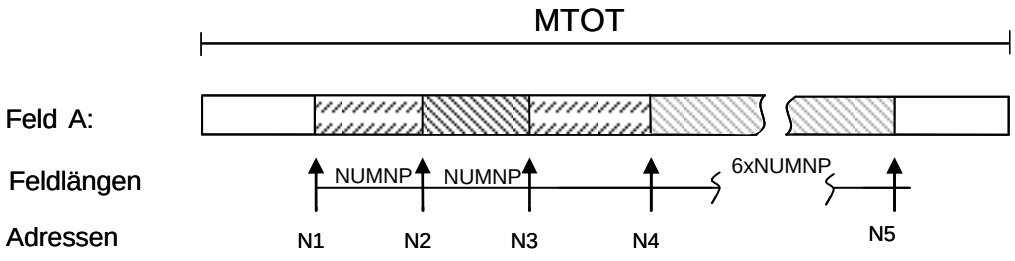


Abb. 4.4-4: Belegung des unbenannten COMMONBLOCKS

Im Hauptprogramm STAN:

Vor der Knoteneingabe:

```
N1 = 601
N2 = N1 + NUMNP
N3 = N2 + NUMNP
```

```
N4 = N3 + NUMNP
N5 = N4 + 6 * NUMNP -1
IF (N5.GT.MTOT)  CALL  ERROR  (N5 – MTOT)
```

Vor der Lasteingabe und Steifigkeitsbildung **K**

`N2 = N1 + NEQ * NUMLC` - rechte Seite $\mathbf{P}_{Lastfall1}$, $\mathbf{P}_{Lastfall2}$.,.....

`N3 = N2 + NEQ * MBAND` - Struktursteifigkeit **K**

Vor der Gleichungslösung

```
NN = N1 + NEQ * (L-1)
                 |
                 Lastfallnummer
```

Nach der Gleichungslösung

`NN = N1+NEQ * (L-1)` - Verschiebungsvektoren $\mathbf{u}_{Lastfall1}$, $\mathbf{u}_{Lastfall2}$.,.....

ii) **Übergabe der Feldlängen in Unterprogramme und Neudimensionierung im Unterprogramm**

siehe Beispiel: „program matrixadd"; Kapitel 2.5 ii) „Platzhalter" in FORTRAN 77

iii) **Volldynamische Felder in FORTRAN90**

Anlegen von variablen Feldern während des Programmlaufs mit dem Befehl „ALLOCATABLE" – siehe Beispiel im Kapitel 2.5 iii)

4.4.6.3 Gleichungslöser SYMSOL

Gleichungslöser für symmetrische Koeffizientenmatrixen mit LDL^T-Zerlegung, Vorwärtsreduzieren und Rückwärtseinsetzen.

Siehe Kap. 3.4

```
      SUBROUTINE SYMSOL(A,B,NN,MM,KKK)
      DIMENSION A(NN,MM,B(NN)
      COMMON /IOFIL/ NT1,NT2,NT3,NIN,NOT
C
      GO TO (1000,2000),KKK
C-----REDUCE-MATRIX
 1000 WRITE (NOT,3000)
      DO 280 N=1,NN
      IF(A(N,1).LE.0.0) WRITE (NOT,3001) N,A(N,1)
      DO 260 L=2,MM
      C=A(N,L)/A(N,1)
      I=N+L-1
      IF(NN-I) 260,240,240
  240 J=0
      DO 250 K=L,MM
      J=J+1
  250 A(I,J)=A(I,J)-C*A(N,K)
  260 A(N,L)=C
```

```
  280 CONTINUE
      GO TO 500
C-----REDUCE VECTOR
 2000 DO 290 N=1,NN
      DO 285 L=2,MM
      I=N+L-1
      IF(NN-I) 290,285,285
  285 B(I)=B(I)-A(N,L)*B(N)
  290 B(N)=B(N)/A(N,1)
C-----BACK SUBSTITUTION
      N=NN
  300 N=N-1
      IF(N) 350,500,350
  350 DO 400 K=2,MM
      L=N+K-1
      IF(NN-L) 400,370,370
  370 B(N) = B(N) - A(N,K)*B(L)
  400 CONTINUE
      GO TO 300
C
  500 RETURN
 3000 FORMAT (/' START OF SOLUTION OF EQUATIONS')
 3001 FORMAT (1X,' EQUATION # -',I5,3X,' DIAGONAL TERM = ',1E15.7/)
      END
```

4.4.6.4 Out-of-core solver

Das Lernprogramm STAN zeigt die typischen Merkmale eines kommerziellen Rechencodes mit Nutzung externer Dateien zum zeitweisen Auslagern nicht benötigter Informationen im Hauptspeicher. Die ausgelagerten Daten können bei Bedarf in die Zentraleinheit (CPU) jederzeit wieder eingelesen werden. Diese Progrmmstrukturen des Lösens erlaubt sehr großen Strukturen (z. B. Karosserieberechnung des Fahrzeugbaus $>10^7$ Knoten oder Elemente) bei vergleichsweise kleinem Hauptspeicher zu berechnen.

5 Balkenelemente

5.1 Balkenelemente nach der EULER-BERNOULLI-Theorie

5.1.1 Gleichgewicht, Kinematik und Elastizitätsmodell

Die Beziehungen für das Gleichgewicht, die Kinematik und die elastischen Materialgleichungen werden getrennt für Stabdehnung und Stabbiegung nach der EULER-BERNOULLI-Theorie zusammengestellt.

Dehnung:

$$\left.\begin{aligned} N' + n &= 0 \\ N &= EA\,\varepsilon \\ \varepsilon &= u' \end{aligned}\right\} \qquad \text{Verschiebungsgleichung:} \qquad (EAu')' + n = 0$$

Biegung:

$$\begin{aligned} M' &= Q \\ Q' &= -q \end{aligned}$$

$$\left.\begin{aligned} M'' + q &= 0 \\ M &= EI\kappa \\ \kappa &= -w'' \end{aligned}\right\} \qquad \text{Verschiebungsgleichung:} \qquad -(EIw'')'' + q = 0$$

Die Querschnittsverdrehung $\varphi(x)$ ist gemäß der Normalenhypothese bei der EULER-BERNOULLI-Theorie gleich der negativen Ableitung der Durchbiegung $w(x)$ - siehe Kapitel 4.1.2:

$$\varphi(x) = -w'(x)$$

5.1.2 Prinzip der virtuellen Verschiebung für den Balken

Aus den Verschiebungsdifferentialgleichungen wird die schwache Form (Integralform) der Felder für die Verschiebungen konstruiert.

Das Prinzip der virtuellen Verschiebungen (P.v.V.) wird für den Balken durch Konstruktion der Integralgleichung für die Lösung der Verschiebungsfelder $u(x), w(x)$ hergeleitet:

- Multiplikation der Verschiebungsdifferentialgleichungen mit den virtuellen Verschiebungen $\delta u(x)$ und $\delta w(x)$
- Addition der beiden Gleichungen
- Integration der Funktionen über die Stablänge x

$$\int_l \left\{ \delta u \left[(EAu')' + n \right] + \delta w \left[-(EIw'')'' + q \right] \right\} \mathrm{d}x = 0$$

Partielle Integration zur Übertragung der Ableitung $\frac{\mathrm{d}}{\mathrm{d}x}(\ ) = (\ )'$ auf die virtuellen Verschiebungen $\delta u(x)$ und $\delta w(x)$:

$$\Leftrightarrow \qquad \int_l \left\{-\delta u'\left(EAu'\right) + \delta w'\left(EIw''\right)'\right\}\mathrm{d}x = \left[-\delta u\left(EAu'\right) + \delta w\left(EIw''\right)'\right]_{x=0}^{x=l} + \int_0^l \left(-\delta u\, n - \delta w\, q\right)\mathrm{d}x$$

Nochmalige partielle Integration des Produkts für den Biegeanteil:

$$\Leftrightarrow \qquad -\int_l \left[\delta u'\left(EAu'\right) - \delta w''\left(EIw''\right)\right]\mathrm{d}x = -\left[\delta u\left(EAu'\right)\right]_{x=0}^{x=l} + \left[\delta w\left(EIw''\right)'\right]_{x=0}^{x=l} - \left[\delta w'\left(EIw''\right)\right]_{x=0}^{x=l} - \int_0^l \left(\delta u\, n + \delta w\, q\right)\mathrm{d}x$$

Wegen des Gleichgewichts zwischen den äußeren Lasten P_x, M_e, P_z und den inneren Kräften N,M,Q an den Balkenenden $x = 0$ oder $x = l$ ergeben sich aus den statischen Randbedingungen folgende Beziehungen:

Beispielhaft gilt am Stabende bei $x = l$:

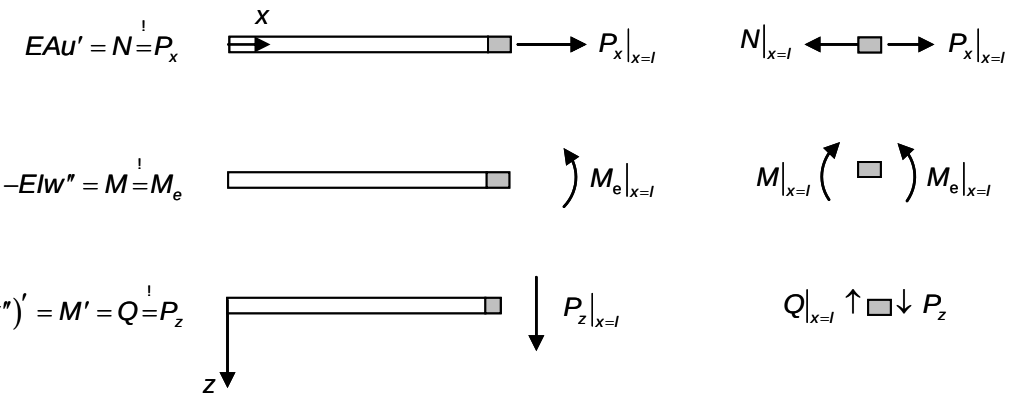


Abb. 5.1-1: Statische Randbedingungen

Sinngemäß wird am Balkenanfang bei $x = 0$ verfahren.

Die Lasten P_x, M_e und P_z werden in die Randterme der Integralgleichung substituiert.

Desweiteren wird die (negative) Ableitung der virtuellen Querverschiebung $\delta w'$ durch die virtuelle Verdrehung $\delta\varphi$ im Term der virtuellen Arbeit für das eingeprägte Moment M_e ersetzt. Dann folgt daraus die schwache Form der Verschiebungsgleichungen für die EULER-BERNOULLI Balkentheorie.

$$\int_l \left[\delta u'\left(EAu'\right) + \delta w''\left(EIw''\right)\right]\mathrm{d}x = \left[\delta u\, P_x\right]_{x=0}^{x=l} + \left[\delta w\, P_z\right]_{x=0}^{x=l} + \left[\delta\varphi\, M_e\right]_{x=0}^{x=l} + \int_l \left(\delta u\, n + \delta w\, q\right)\mathrm{d}x$$

Das P.v.V ergibt sich daraus, falls folgende Beziehungen eingesetzt werden:

$$EAu' = N \qquad \text{und} \qquad -EIw'' = M$$

$$\text{sowie} \qquad u' = \varepsilon \qquad \text{und} \qquad w'' = -\kappa$$

als auch $\delta u' = \delta\varepsilon$ und $\delta w'' = -\delta\kappa$

P.v.V.:

$$\int_l \left(\delta\varepsilon N + \delta\kappa M\right)\mathrm{d}x = \delta u P_x\big|_{x=x_{Px}} + \delta w P_z\big|_{x=x_{Pz}} + \delta\varphi M_e\big|_{x=x_{Me}} + \int_l \left(\delta u n + \delta w q\right)\mathrm{d}x$$

wobei $x = x_{Px}$, $x = x_{Pz}$ und $x = x_{Me}$ die jeweilige Position der Einzellasten am Kräfterand bedeuten.

Am Verschiebungsrand bei

$x = x_u$ bzw. $x = x_w$ sowie $x = x_\varphi$

müssen die virtuellen Verschiebungen verschwinden, sodass gilt:

$$\delta u\big|_{x=x_u} = 0, \qquad \delta\varphi\big|_{x=x_u} = 0 \quad \text{oder} \quad \delta w\big|_{x=x_w} = 0$$

Die virtuelle innere Arbeit ist demgemäß definiert zu

$$\delta A_\mathrm{i} = -\int_l \left(\delta\varepsilon N + \delta\kappa M\right)\mathrm{d}x$$

und die virtuelle Arbeit der äußeren Lasten ist

$$\delta A_\mathrm{a} = \delta u P_x\big|_{x=x_{Px}} + \delta w P_z\big|_{x=x_{Pz}} + \delta\varphi M_e\big|_{x=x_{Me}} + \int_l \left(\delta u n + \delta w q\right)\mathrm{d}x$$

Die gesamte virtuelle Arbeit ergibt sich zu:

$$\delta A_\mathrm{ges} = \delta A_\mathrm{i} + \delta A_\mathrm{a} = 0$$

Satz: Das P.v.V. lautet demgemäß:

„Die gesamte virtuelle Arbeit des Balkens muss verschwinden“

$$\delta A_\mathrm{ges} = 0$$

Die Einführung des kinematischen Operators **L** (linearer Differentialoperator):

$$\begin{bmatrix}\varepsilon\\ \kappa\end{bmatrix} = \begin{bmatrix}u'\\ -w''\end{bmatrix} = \begin{bmatrix}\dfrac{\mathrm{d}}{\mathrm{d}x} & 0\\ 0 & -\dfrac{\mathrm{d}^2}{\mathrm{d}x^2}\end{bmatrix}\begin{bmatrix}u\\ w\end{bmatrix} =: \mathbf{Lu}$$

erlaubt, die schwache Form des EULER-BERNOULLI-Stabs in Matrizenschreibweise anzugeben:

$$\int_l \begin{bmatrix}\delta u & \delta w\end{bmatrix}\begin{bmatrix}\dfrac{\mathrm{d}}{\mathrm{d}x} & 0\\ 0 & -\dfrac{\mathrm{d}^2}{\mathrm{d}x^2}\end{bmatrix}\begin{bmatrix}EA & 0\\ 0 & EI\end{bmatrix}\begin{bmatrix}\dfrac{\mathrm{d}}{\mathrm{d}x} & 0\\ 0 & -\dfrac{\mathrm{d}^2}{\mathrm{d}x^2}\end{bmatrix}\begin{bmatrix}u\\ w\end{bmatrix}\mathrm{d}x = \delta u P_x\big|_{x=x_{Px}} + \delta w P_z\big|_{x=x_{Pz}} + \delta\varphi M_e\Big|_{x=x_{Me}} + \int_l \begin{bmatrix}\delta u & \delta w\end{bmatrix}\begin{bmatrix}n\\ q\end{bmatrix}\mathrm{d}x$$

Einführung der Vektorschreibweise für die Feldlasten in Längs- und Querrichtung:

$$\mathbf{p}_\mathrm{l} := \begin{bmatrix}n(x)\\ q(x)\end{bmatrix};$$

und für die Randkräfte:

$$\mathbf{P}_l := \begin{bmatrix} P_x \\ P_z \end{bmatrix}$$

Die Steifigkeiten EA und EI werden in der Matrix **C** zusammengefasst.

$$\mathbf{C} = \begin{bmatrix} EA & 0 \\ 0 & EI \end{bmatrix}$$

Die beiden Feldgleichungen $u(x)$ und $w(x)$ werden mit dem Vektor

$$\mathbf{u}(x) = \begin{bmatrix} u(x) \\ w(x) \end{bmatrix}$$

abgekürzt.

Dann lautet die schwache Form der Verschiebungsgleichungen unter Hinzunahme der Elastizitätsbeziehungen für den EULER-BERNOULLI-Stab in symbolischer Notation:

$$\underbrace{\int_l \delta\mathbf{u}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{C}\mathbf{L}\,\mathbf{u}\,\mathrm{d}x}_{\delta W} = \underbrace{\delta\mathbf{u}^{\mathrm{T}}\mathbf{P}_l\Big|_{x=x_p} + \delta\varphi M_e\Big|_{x=x_{Me}}}_{\text{virtuelle Arbeit der Randlasten}} + \underbrace{\int_l \delta\mathbf{u}^{\mathrm{T}}\mathbf{p}_l\,\mathrm{d}x}_{\substack{\text{virtuelle Arbeit} \\ \text{der Feldlasten}}}$$

Die linke Gleichungsseite stellt die virtuelle Änderung δW der Formänderungsenergie W des elastischen Balkens unter Längsdehnung und Biegung dar.

Satz: „Die virtuelle Arbeit der äußeren Kräfte ist gleich der virtuellen Änderung der Formänderungsenergie“.

$$\delta W = \delta A_{\mathrm{a}}$$

5.1.3 Diskretisierung der Verschiebungsfelder für die Längs- und Querrichtung

5.1.3.1 Ansatz für die Längsverschiebung

In den Integralgleichungen für die schwache Form treten erste Ableitungen $\frac{\mathrm{d}}{\mathrm{d}x}$ der Längsverschiebung $u(x)$ auf.

Hinweis:

- Deshalb ist C°-Stetigkeit des Verschiebungsansatzes $u_{\mathrm{h}}(x)$ im Element und über die Elementzwischengrenzen hinweg ausreichend.
- Eine lineare Verschiebungsinterpolation $u_{\mathrm{h}}(x)$ im Element wird in Stablängsrichtung angenommen.
- Die Stetigkeit des Verschiebungsfeldes $u_{\mathrm{h}}(x)$ an den Elementgrenzen ist notwendig.

Die Freiheitsgrade des Dehnstabelements $\boxed{k}$ für die Längsverschiebung sind die Knotenverschiebungen u_i und u_j an den Enden.

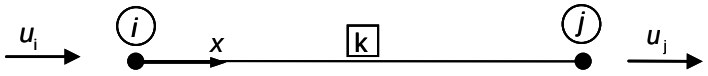


Abb. 5.1-2: Dehnstabelement mit zwei Freiheitsgraden

Einführung der Einheitskoordinate $\xi = \frac{x}{l}$.

Der Verschiebungsverlauf $u_h(x)$ für die Längsrichtung wird linear angenommen und soll durch die Formfunktionen $N_1(\xi)$ und $N_4(\xi)$ beschrieben werden.

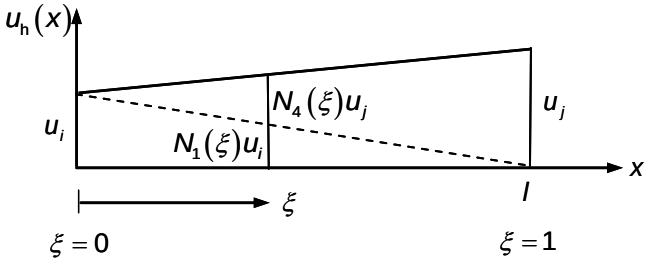


Abb. 5.1-3: Ansatz für den Verschiebungsverlauf $u_h(x)$ im linearen Dehnstabelement

Vektor der Knotenverschiebung des Dehnstabs $\mathbf{u}_D = \begin{bmatrix} u_i \\ u_j \end{bmatrix}$

Verschiebungsinterpolation:

$$u_h(x) = N_1(\xi)\, u_i + N_4(\xi)\, u_j = \mathbf{N}_D(\xi)\, \mathbf{u}_D$$

mit den Formfunktionen:

$$N_1(\xi) = 1 - \xi$$

$$N_4(\xi) = \xi,$$

die in der Matrix der Formfunktion $\mathbf{N}_D$ abgelegt sind.

Wenn keine Feldlasten $n(x) \equiv 0$ auftreten, ist der lineare Verschiebungsansatz u_h identisch gleich der exakten Lösung $u(x)$:

$$u(x) \equiv u_h(x) = \left(1 - \frac{x}{l}\right) u_i + \frac{x}{l} u_j$$

$$\varepsilon_h(x) = u_h'(x) = -\frac{1}{l} u_i + \frac{1}{l} u_j = \begin{bmatrix} -\frac{1}{l} & \frac{1}{l} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} =: \mathbf{B}_D\, \mathbf{u}_D$$

Die virtuellen Verschiebungs- $\delta u_h(x)$ und virtuelle Dehnungsverläufe

$\delta\varepsilon_h(x)$ ergeben sich aus den virtuellen Knotenverschiebungen δu_i und δu_j:

$$\delta u_h(x) = N_1(x)\ \delta u_i + N_4(x)\ \delta u_j$$

$$\delta u_h'(x) = -\frac{1}{l}\ \delta u_i + \frac{1}{l}\ \delta u_j$$

$$\delta\varepsilon_{\mathrm{h}}(x) = -\frac{1}{l}\delta u_i + \frac{1}{l}\delta u_j = \begin{bmatrix} -\frac{1}{l} & \frac{1}{l} \end{bmatrix} \begin{bmatrix} \delta u_i \\ \delta u_j \end{bmatrix} =: \mathbf{B}_{\mathrm{D}}\, \delta\mathbf{u}_{\mathrm{D}} \quad ,$$

wobei $\mathbf{B}_{\mathrm{D}}$ die Operatormatrix darstellt.

Die Steifigkeitsmatrix $\mathbf{k}_{\mathrm{D}}$ des Dehnstabelements (Fachwerkelements) folgt aus der schwachen Form (Variationsgleichung) für den Anteil der Längsdehnung:

$$\int_l \delta u_{\mathrm{h}}' \left(EA\, u_{\mathrm{h}}' \right) \mathrm{d}x = \int_l \delta\mathbf{u}_{\mathrm{D}}^{\mathrm{T}}\, \mathbf{B}_{\mathrm{D}}^{\mathrm{T}}\, EA\, \mathbf{B}_{\mathrm{D}}\, \mathbf{u}_{\mathrm{D}}\, \mathrm{d}x$$

$$= \delta\mathbf{u}_{\mathrm{D}}^{\mathrm{T}} \underbrace{\int_l \mathbf{B}_{\mathrm{D}}^{\mathrm{T}}\, EA\, \mathbf{B}_{\mathrm{D}}\, \mathrm{d}x}_{=:\mathbf{k}_{\mathrm{D}}}\, \mathbf{u}_{\mathrm{D}} \qquad \text{Elementsteifigkeit des Dehnstabs}$$

Einsetzen des Verschiebungsansatzes $u_{\mathrm{h}}(x)$:

$$\mathbf{k}_{\mathrm{D}} = \int_l \begin{bmatrix} -\frac{1}{l} \\ \frac{1}{l} \end{bmatrix} EA \begin{bmatrix} -\frac{1}{l} & \frac{1}{l} \end{bmatrix} \mathrm{d}x = \int_l \begin{bmatrix} \frac{1}{l^2} & -\frac{1}{l^2} \\ -\frac{1}{l^2} & \frac{1}{l^2} \end{bmatrix} EA\, \mathrm{d}x$$

Die Elementsteifigkeit für den Dehnstab mit linearem Verschiebungsansatz $u_{\mathrm{h}}(x)$ lautet - siehe auch Skriptum zur Vorlesung FEM:

$$\mathbf{k}_{\mathrm{D}} = \frac{EA}{l} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

5.1.3.2 Ansatz für die Querverschiebung

In den Integralgleichungen der schwachen Form (Variationsgleichungen) treten zweite Ableitungen $\frac{\mathrm{d}^2}{\mathrm{d}x^2}$ der Querverschiebung $w(x)$ auf.

Hinweise:

- C^1- Stetigkeit ist an den Elementzwischengrenzen für die Interpolationsfunktion der Querverschiebung w notwendig.
- Kubische Verschiebungsansätze mit 4 Parametern sind notwendig, um die Querverschiebungen **w** und deren Ableitung $w' = -\varphi$ an den beiden Elementenden zu kontrollieren.
- Anstelle von Zwischenknoten ist es günstiger, direkt die Biegelinienneigung $w'(x)$ an den Stabenden durch Knotendrehwinkel φ_{i} und φ_{j} zu beschreiben. Deshalb werden an den Stabenden zusätzlich die Knotendrehwinkel φ_{i} und φ_{j} als Unbekannte neben den Querverschiebungen w_{i} und w_{j} eingeführt.

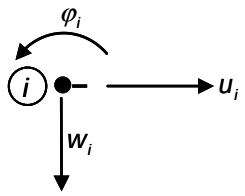


Abb. 5.1-4: Knotenfreiheitsgrade des Biegestabelements

Der Verschiebungsansatz für die Querrichtung wird in Form eines kubischen Polynoms angesetzt.

$$w_{\mathrm{h}}(x) = \tilde{c}_0 + \tilde{c}_1 x + \tilde{c}_2\, x^2 + \tilde{c}_3\, x^3$$

Für die Herleitung des Ansatzes $w_{\mathrm{h}}(x)$ der Biegelinie wird der Einfachheit halber mit der normierten Stablängskoordinate $\xi = \frac{x}{l}$ gearbeitet:

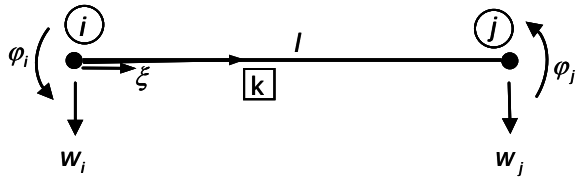


Abb. 5.1-5: Elementfreiheitsgrade des Biegestabelements

Verschiebungsansatz:

$$w_{\mathrm{h}}(\xi) = c_0 + c_1\xi + c_2\xi^2 + c_3\xi^3$$

Die Polynomkoeffizienten c_0, c_1, c_2 und c_3 werden in Abhängigkeit der Knotenfreiheitsgrade w_i, φ_i und w_j, φ_j des Elements bestimmt.

$$\mathbf{u_B} := \begin{bmatrix} w_{\mathrm{i}} \\ \varphi_{\mathrm{i}} \\ w_{\mathrm{j}} \\ \varphi_{\mathrm{j}} \end{bmatrix}$$

Aus den Randbedingungen $w\big|_{\xi=0}, w\big|_{\xi=1}$ sowie $w'\big|_{\xi=0}$ und $w'\big|_{\xi=1}$ folgt das System von Gleichungen zur Berechnung der Polynomkoeffizienten.

$$\begin{bmatrix} w\big|_{\xi=0} \\ -w'\big|_{\xi=0} \\ w\big|_{\xi=1} \\ -w'\big|_{\xi=1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{l} & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & -\frac{1}{l} & -\frac{2}{l} & -\frac{3}{l} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} w_i \\ \varphi_i \\ w_j \\ \varphi_j \end{bmatrix}$$

Nach der Inversion der Koeffizientenmatrix ergeben sich die Polynomkoeffizienten c_0, c_1, c_2 und c_3 in Abhängigkeit der Knotenfreiheitsgrade $w_{\mathrm{i}}, \varphi_{\mathrm{i}}, w_{\mathrm{j}}$ und φ_{j} zu:

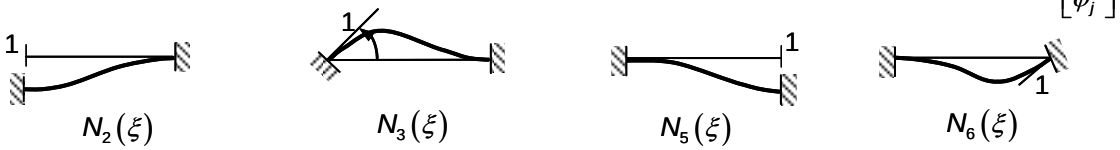
$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -l & 0 & 0 \\ -3 & 2l & 3 & l \\ 2 & -l & -2 & -l \end{bmatrix} \begin{bmatrix} w_\mathrm{i} \\ \varphi_\mathrm{i} \\ w_\mathrm{j} \\ \varphi_\mathrm{j} \end{bmatrix}$$

Die Koeffizienten werden in den Polynomansatz $w_\mathrm{h}(\xi)$ eingesetzt; der Ansatz wird damit in den Knotenverschiebungsgrößen ausgedrückt:

$$w(\xi) = \left(1-3\xi^2+2\xi^3\right)w_\mathrm{i} + l\,\xi\left(-1+2\xi-\xi^2\right)\varphi_\mathrm{i} + \xi^2\left(3-2\xi\right)w_j + l\,\xi^2\left(1-\xi\right)\varphi_j$$

Der Verschiebungsansatz kann als Produkt eines Zeilenvektors mit einem Spaltenvektor geschrieben werden. Die Elemente des Zeilenvektors beschreiben den Verlauf der Biegelinie $w(x)$ in Abhängigkeit des Einflusses eines Knotenfreiheitsgrades gleich „eins“ und der Festhaltung aller anderen drei Freiheitsgrade. Diese Verformungen bezeichnet man als Einheitsverschiebungszustände und nennt die dazugehörigen Funktionsgleichungen die Formfunktionen $N_2(\xi), N_3(\xi), N_5(\xi), N_6(\xi)$ des Balkenelements nach der EULER-BERNOULLI-Theorie.

$$w_\mathrm{h}(\xi) = \left[\left(1-3\xi^2+2\xi^3\right) \quad l\xi\left(-1+2\xi-\xi^2\right) \quad \xi^2\left(3-2\xi\right) \quad l\xi^2\left(1-\xi\right)\right] \cdot \begin{bmatrix} w_i \\ \varphi_i \\ w_j \\ \varphi_j \end{bmatrix}$$



$N_2(\xi)$ $N_3(\xi)$ $N_5(\xi)$ $N_6(\xi)$

Abb. 5.1-6: Einheitsverschiebungszustände (Formfunktionen) des Biegestabs

$$w_\mathrm{h}(\xi) = \mathbf{N}_\mathrm{B}\,\mathbf{u}_\mathrm{B} = \left[N_2(\xi) \quad N_3(\xi) \quad N_5(\xi) \quad N_6(\xi)\right] \begin{bmatrix} w_\mathrm{i} \\ \varphi_\mathrm{i} \\ w_\mathrm{j} \\ \varphi_\mathrm{j} \end{bmatrix}$$

Die vier kubischen Funktionen $N_i(x)$ werden auch Hermite-Polynome genannt. Dieser Ansatz wird in die geometrische Gleichung eingesetzt und damit die Operatormatix $\mathbf{B}_\mathrm{B}$ ermittelt. Wegen der zweimaligen Ableitung enthält sie hier nur lineare Terme in ξ bzw. **x**.

Kinematische Gleichung für die Krümmung des Biegestabs:

$$\kappa(x) = -w''(x)$$

$$= -\mathbf{N}_\mathrm{B}'' \cdot \mathbf{v} = -\frac{1}{l^2}\,\frac{\mathrm{d}^2\mathbf{N}_B}{\mathrm{d}\xi^2}\,\mathbf{u}_\mathrm{B}$$

$$= \underbrace{-\left[\frac{1}{l^2}\left(-6+12\xi\right) \quad \frac{1}{l}\left(4-6\xi\right) \quad \frac{1}{l^2}\left(6-12\xi\right) \quad \frac{1}{l}\left(2-6\xi\right)\right]}_{\mathbf{B}_\mathrm{B}(\xi) \quad \text{Operatormatrix}} \begin{bmatrix} w_\mathrm{i} \\ \varphi_\mathrm{i} \\ w_\mathrm{j} \\ \varphi_\mathrm{j} \end{bmatrix}$$

$$= \mathbf{B}_\mathrm{B}\,\mathbf{u}_\mathrm{B}$$

5.1.4 Steifigkeitsmatrix des Biegeelements nach der EULER-BERNOULLI-Theorie

Der Ausdruck für die virtuelle Änderung der Formänderungsenergie $\int \delta w''(EIw'')\mathrm{d}x$ des Biegestabs liefert mit dem Ansatz für die Diskretisierung der Querverschiebung $w(x)$ bzw. deren virtuellen Verlauf $\delta w(x)$

$$\delta \mathbf{u}_\mathrm{B}^\mathrm{T} \int_0^l \mathbf{B}_\mathrm{B}^\mathrm{T}\, EI\, \mathbf{B}_\mathrm{B}\, \mathrm{d}x\, \mathbf{u}_\mathrm{B} =: \delta \mathbf{u}_\mathrm{B}^\mathrm{T}\ \mathbf{k}_\mathrm{B}\ \mathbf{u}_\mathrm{B}$$

die Steifigkeitsmatrix des Biegeelements in den lokalen Koordinaten:

$$\mathbf{k}_\mathrm{B} = l\, EI \int_0^1 \mathbf{B}_\mathrm{B}^\mathrm{T}(\xi)\, \mathbf{B}_\mathrm{B}(\xi)\, \mathrm{d}\xi$$

Für den Integranden ergibt sich das Matrizenprodukt aus dem FALKschen Anordnungsschema:

$\mathbf{B}_\mathrm{B}^\mathrm{T}$ \ $\mathbf{B}_\mathrm{B}$ ⊖	$\frac{1}{l^2}(-6+12\xi)$	$\frac{1}{l}(4-6\xi)$	$\frac{1}{l^2}(6-12\xi)$	$\frac{1}{l}(2-6\xi)$
$\frac{1}{l^2}(-6+12\xi)$	$\frac{36}{l^4}(2\xi-1)^2$	⊔⊔	⊔⊔	⊔⊔
$\frac{1}{l}(4-6\xi)$		$\frac{4}{l^2}(2-3\xi)^2$	⊔⊔	⊔⊔
$\frac{1}{l^2}(6-12\xi)$	symm.		$\frac{36}{l^4}(1-2\xi)^2$	⊔⊔
$\frac{1}{l}(2-6\xi)$				$\frac{4}{l^2}(1-3\xi)^2$

Abb. 5.1-7: Produkt $\mathbf{B}_\mathrm{B}^\mathrm{T}\mathbf{B}_\mathrm{B}$ des diskretisierten, kinematischen Operators

Die analytische Integration des Biegeanteils für den Stab liefert folgende Steifigkeitskomponenten:

$$k_{11} = l\, EI \frac{36}{l^4}\int_0^1 (2\xi-1)^2 d\xi = 36\frac{EI}{l^3}\left[\frac{1}{2}\frac{1}{3}(2\xi-1)^3\right]_0^1 = 12\frac{EI}{l^3}$$

$$k_{22} = l\, EI \frac{4}{l^2}\int_0^1 (3\xi-2)^2 d\xi = 4\frac{EI}{l}\left[\frac{1}{3}(3\xi-2)^3\frac{1}{3}\right]_0^1 = 4\frac{EI}{l}$$

$$\vdots$$

Das Ergebnis obiger Rechnung ist die Elementsteifigkeit $\mathbf{k}_\mathrm{B}$ für die Biegung des Stabs in lokalen Koordinaten $\mathbf{u}_\mathrm{B}$:

$$\begin{bmatrix} Q_i \\ M_i \\ Q_j \\ M_j \end{bmatrix} = \frac{EI}{l} \begin{bmatrix} \frac{12}{l^2} & -\frac{6}{l} & -\frac{12}{l^2} & -\frac{6}{l} \\ & 4 & \frac{6}{l^2} & 2 \\ & \text{symm.} & \frac{12}{l^2} & \frac{6}{l} \\ & & & 4 \end{bmatrix} \cdot \begin{bmatrix} w_i \\ \varphi_i \\ w_j \\ \varphi_j \end{bmatrix}$$

In symbolischer Notation:

$$\mathbf{S}_B = \mathbf{k}_B \cdot \mathbf{u}_B$$

bezeichnet $\mathbf{S}_B$ den Vektor für die Momente M_i und M_j sowie die Querkräfte Q_i und Q_j an den Endknoten des Biegeelements. Folgende Regelung ergibt sich dann für die positiven Elementkräfte:

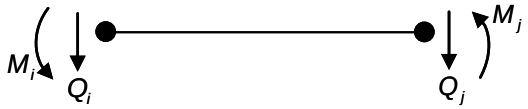


Abb. 5.1-8: Elementschnittgrößen

Die obige Steifigkeitsmatrix des Biegestabs kann durch die Dehnungsanteile mit linearem Ansatz für die Längsverschiebung $u_h(x)$ sinngemäß ergänzt werden. Siehe dazu Kapitel 5.1.3.1.

5.1.5 Erweiterung des Biege- und Dehnstabelements zum ebenen Balkenelement

Im Allgemeinen fällt die Stabachse nicht mit der $x-$Achse $\mathbf{E}_x$ des globalen Koordinatensystems zusammen, mit dem eine Bauteilgruppe berechnet wird. Deshalb muss aus den Gleichungen für den Biege- und Dehnstab das Balkenelement für eine beliebige Lage der Stabachse im Raum durch Koordinatentransformation von lokalen auf globale Verschiebungsfreiheitsgrade hergeleitet werden.

Knotenverschiebungen v_i sind die Freiheitsgrade der Knoten.

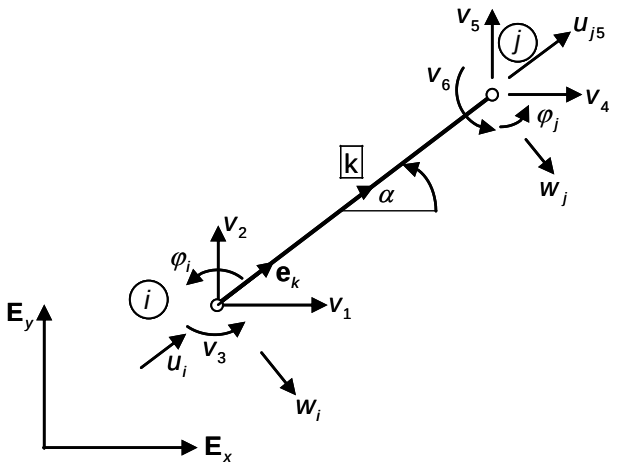
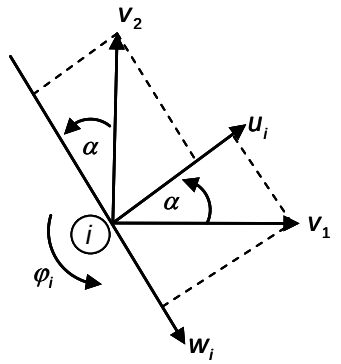


Abb. 5.1-9: Stabelement mit Freiheitsgraden v_1 bis v_6 im Koordinatensystem $\mathbf{E}_x, \mathbf{E}_y$ der Struktur

Die Matrix **a** transformiert von den globalen Freiheitsgraden v_1 bis v_6 des Balkenelements auf die lokalen Freiheitsgrade u_i bis φ_j des Biege- und Dehnstabs.

$$\begin{bmatrix} u_i \\ w_i \\ \varphi_i \\ u_j \\ w_j \\ \varphi_j \end{bmatrix} = \begin{bmatrix} cx & +sx & 0 & 0 & 0 & 0 \\ +sx & -cx & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cx & sx & 0 \\ 0 & 0 & 0 & +sx & -cx & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}$$

$$\mathbf{u}_{D+B} = \mathbf{a} \quad \mathbf{v}$$



Achtung!
Keine reine orthogonale Transformation; wegen der Verschiebung *w* "positiv nach unten" beim Stab ist auch eine mathematische Reflexion an der $x-$Achse des Balkens bei der Transformation notwendig!

Abb. 5.1-10: Lokale Elementfreiheitsgrade und globale Knotenfreiheitsgrade

Für $\alpha = 0^0$ fallen Stabachse und $\mathbf{E}_x -$ Richtung zusammen,

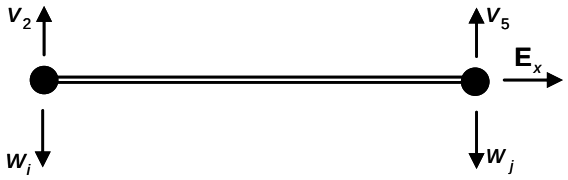


Abb. 5.1-11: Balkenelement

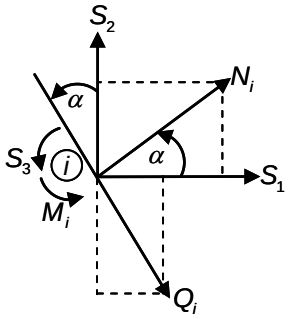
so dass $cx = 1$ und $sx = 0$ ist. Die Transformation zwischen den globalen Freiheitsgraden **v** des Balkenelements und den lokalen Freiheitsgraden $\mathbf{u}_{D+B}$:

$$\mathbf{u}_{D+B} = \mathbf{a} \quad \mathbf{v}$$

lautet dann in ausführlicher Komponentenschreibweise:

$$\begin{bmatrix} u_i \\ w_i \\ \varphi_i \\ u_j \\ \varphi_j \\ w_j \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}$$

Die Elementkräfte im Vektor $\mathbf{S}_{D+B}$ werden vom lokalen Koordinatensystem des Biege- und Dehnstabs in das Koordinatensystem der Struktur $\mathbf{E}_x, \mathbf{E}_y$ zu den Elementkräften **S** entlang der Knotenverschiebungen **v** transformiert.



$$\begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{bmatrix} = \begin{bmatrix} cx & sx & 0 & 0 & 0 & 0 \\ sx & -cx & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cx & sx & 0 \\ 0 & 0 & 0 & sx & -cx & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N_i \\ Q_i \\ M_i \\ N_j \\ Q_j \\ S_j \end{bmatrix}$$

$$\mathbf{S} = \mathbf{a}^T \mathbf{S}_{D+B}$$

Zwischen den Elementkräften wird mit der Transponierten $\mathbf{a}^T$ der Matrix **a** transformiert.

Hinweis: Im Sonderfall des Balkenelements ist $\mathbf{a} = \mathbf{a}^T$

Einsetzen liefert:

$$\mathbf{S} = \mathbf{a}^T \mathbf{k} \mathbf{a} \mathbf{v}$$

Elementkräfte (Schnittgrößen) des Balkens

Die Elementkräfte (Schnittgrößen) des Stabs mit kubischem Verschiebungsansatz für die Biegung und linearem Ansatz für die Längsverformung ergeben sich aus den lokalen Knotenfreiheitsgraden und der lokalen Steifigkeitsmatrix **k**

$$\mathbf{S}_{D+B} = \mathbf{k}\mathbf{u}_{D+B} \; ; \qquad \begin{bmatrix} N_i \\ Q_i \\ M_i \\ N_j \\ Q_j \\ M_j \end{bmatrix} = \begin{bmatrix} \frac{EA}{l} & 0 & 0 & -\frac{EA}{l} & 0 & 0 \\ 0 & \frac{12EI}{l^3} & -\frac{6EI}{l^2} & 0 & -\frac{12EI}{l^3} & -\frac{6EI}{l^2} \\ 0 & & 4\frac{EI}{l} & 0 & \frac{6EI}{l^2} & 2\frac{EI}{l} \\ -EA/l & & & EA/l & 0 & 0 \\ 0 & & & & \frac{12EI}{l^3} & 6\frac{EI}{l^2} \\ 0 & & & & & 4\frac{EI}{l} \end{bmatrix} \begin{bmatrix} u_i \\ w_i \\ \varphi_i \\ u_j \\ w_j \\ \varphi_j \end{bmatrix}$$

Die globale Elementsteifigkeitsmatrix $\mathbf{k}_{global}$ ergibt sich durch Transformation aus der lokalen Steifigkeitsmatrix **k**

$$\mathbf{k}_{global} = \mathbf{a}^T \mathbf{k} \mathbf{a}$$

Hinweis: In der Übung wird ein ingenieuranschaulicher Weg zur Ermittlung der Balkensteifigkeitsmatrix **k** angegeben.

Mit Hilfe der Lokal – Global – Transformation erfolgt die Kraft-Verschiebungsmatrix:

$$\mathbf{S}_{D+B} = \underbrace{\mathbf{k}\,\mathbf{a}}_{\text{Kraft-Verschiebungs-matrix}}\,\mathbf{v}$$

5.1.6 Konsistenter Lastvektor aus Elementbelastung in Längs- und Querrichtung zur Stabachse

Aus dem Lastterm (rechte Seite) folgt die äußere virtuelle Arbeit

$$\delta A_{\mathrm{a}}^{\mathrm{q+n}} = \delta \mathbf{u}^{\mathrm{T}} \underbrace{\int_0^1 \mathbf{N}^{\mathrm{T}} \begin{bmatrix} n \\ q \end{bmatrix} l \,\mathrm{d}\xi}_{=:\mathbf{S}_{\mathrm{q+n}}^0} = \delta \mathbf{u}_{\mathrm{D+B}}^{\mathrm{T}} \mathbf{S}_{\mathrm{q+n}}^0$$

Für die konstante äußere Querlast $q(x) = q_0$ und konstante Längsbelastung $n(x) = n_0$ ergibt sich der Elementlastvektor:

$$\mathbf{S}_{q+n}^0 = +l \int_0^1 \mathbf{N}^{\mathrm{T}} \begin{bmatrix} n \\ q \end{bmatrix} d\xi$$

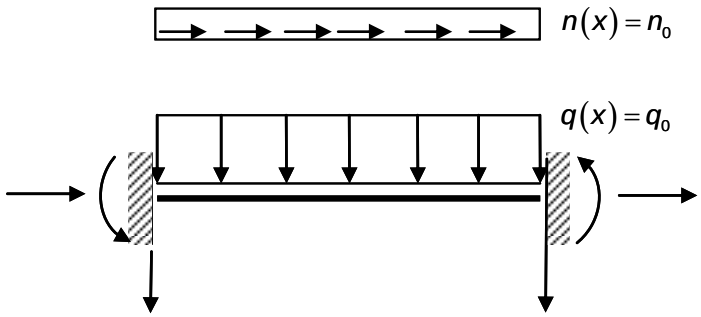


Abb. 5.1-12: Konstante Feldlasten n_0 und q_0 mit Elementlasten $\mathbf{s}_{q+n}^0$

$$\mathbf{p} = \mathbf{a}^{\mathrm{T}} \mathbf{S}_{\mathrm{q+n}}^0 = \mathbf{a}^{\mathrm{T}} \mathbf{S}_{\mathrm{q+n}}^0 = \mathbf{a}^{\mathrm{T}} l \int_0^1 \begin{bmatrix} 1-\xi & 0 \\ 0 & 1-3\xi^2+2\xi^3 \\ 0 & l\xi\left(-1+2\xi-\xi^2\right) \\ \xi & 0 \\ 0 & \xi^2(3-2\xi) \\ 0 & l\xi^2(1-\xi) \end{bmatrix} \begin{bmatrix} \mathbf{n} \\ \mathbf{q} \end{bmatrix} \mathrm{d}\xi = + \int_0^1 \begin{bmatrix} n_0 l(1-\xi) \\ q_0 l^2 \left(1-3\xi^2+2\xi^3\right) \\ q_0 l \xi\left(-1+2\xi-5\xi^2\right) \\ n_0(l+\xi) \\ q_0 l \ \xi^2(3-2\xi) \\ q_0 l^2 \xi^2(1-\xi) \end{bmatrix} \mathrm{d}\xi = \begin{bmatrix} +\frac{n_0 l}{2} \\ +\frac{q_0 l}{2} \\ -\frac{q_0 l^2}{12} \\ +\frac{n_0 l}{2} \\ +\frac{q_0 l}{2} \\ +\frac{q_0 l^2}{12} \end{bmatrix}$$

Anhand der Transformation des lokalen Elementlastvektors $\mathbf{S}_{\mathrm{q+n}}^0$ auf die Strukturfreiheitsgrade **v** im globalen Koordinatensystem mit Hilfe von:

$$\mathbf{u}_{\mathrm{D+B}} = \mathbf{a} \quad \mathbf{v}$$

folgt der Elementlastvektor **p** aus der virtuellen äußeren Arbeit mit den virtuellen Verschiebungen $\delta\mathbf{v}$ der Strukturfreiheitsgrade:

$$\delta A_{\mathrm{a}}^{\mathrm{q+n}} = \delta \mathbf{v}^{\mathrm{T}} \ \mathbf{a}^{\mathrm{T}} \int_0^l \mathbf{N}^{\mathrm{T}} \begin{bmatrix} n \\ q \end{bmatrix} l \,\mathrm{d}\xi = \delta \mathbf{v}^{\mathrm{T}} \underbrace{\mathbf{a}^{\mathrm{T}} \mathbf{S}_{\mathrm{q+n}}^0}_{\mathbf{p}}$$

Definition des Elementlastvektors **p** in globalen Koordinaten:

$$\mathbf{p} = \mathbf{a}^{\mathrm{T}} \ \mathbf{S}_{\mathrm{q+n}}^0$$

5.2 Balkenelemente nach der TIMOSHENKO – Theorie

5.2.1 Grundgleichungen

Kurze Biegestäbe und Sandwichbalken zeigen unter Querbelastung neben der Krümmung auch noch merkliche Schubverformungen. Die Balkentheorie von TIMOSHENKO erfasst die Schubverformungen zwischen Stabachse und Querschnitt.

Dehnung: - wie EULER – BERNOULLI – Balkenelement

Die Biegung wird durch die folgenden bereits oben erwähnten drei Gleichungsgruppen beschrieben:

GGW:

$$\left.\begin{aligned} M' &= Q \\ Q' &= -q \end{aligned}\right\} \qquad M'' + q = 0 \tag{5.2.1}$$

Kinematik:

$$\varepsilon_x = \varphi'(x) \cdot z \tag{5.2.2}$$

$$\gamma_{xz} = w' + \varphi \tag{5.2.3}$$

$$\kappa = \varphi' \tag{5.2.4}$$

Elastizität: $\sigma_x = E\ \varepsilon_x$:

$$M(x) = \int_{(A)} \sigma_x\, z\, dA = \int_{(A)} E\varepsilon_x\, z\, dA = \int_{(A)} E\left(\varphi'(x) z\right) z\, dA$$

$$M(x) = E\varphi' \underbrace{\int_{(A)} z^2\, dA}_{=I}$$

$$M(x) = EI\varphi'(x) \tag{5.2.5}$$

Einsetzen von Gl. (5.2.4) in Gl. (5.2.5) liefert für die Biegung:

$$M(x) = EI\kappa(x) \tag{5.2.6}$$

$$\tau_{xz} = G\gamma_{xz} : \; Q(x) = \int_{(A)} \tau_{xz}\, dA = GA_s\ \gamma_{xz}$$

und für den für Schub:

$$\boxed{Q(x) = GA_s\ \gamma_{xz}} \tag{5.2.7}$$

Einsetzen von Gl. (5.2.3) in Gl. (4.6.7):

$$Q(x) = GA_s\left(w'(x) + \varphi(x)\right) \tag{5.2.8}$$

5.2.2 Verschiebungsgleichungen des Biegestabs nach der TIMOSHENKO-Theorie

Durch Einsetzen der Elastizitätsgleichungen für die Biegung und den Schub in die Gleichgewichtsbeziehungen werden die Schnittgrößen **M** und **Q** eliminiert.

Gl. (5.2.5) in Gleichgewichtsbedingung (5.2.1) $\boxed{\left[EI\varphi'(x)\right]'' = -q(x)}$ (5.2.9)

Gl. (5.2.8) in Gleichgewichtsbedingung $Q' = -q$: $\left[GA_s\left(w'(x)+\varphi(x)\right)\right]' = -q(x)$

Falls der Querschnittsverlauf unveränderlich ist, d.h. $GA_s = \text{konst}$ in x, dann folgt:

$$w''(x) = -\varphi'(x) - \frac{q(x)}{GA_s} \tag{5.2.10}$$

oder: Gl. (5.2.5) und (5.2.8) in $M' = Q$ einsetzen:

$$\left[EI\varphi'\right]' = GA_s\left(w'(x)+\varphi(x)\right)$$

Falls die Biegesteifigkeit $EI = \text{konst}$ in x ist, gilt :

$$w' = \frac{EI}{GA_s}\varphi'' - \varphi(x) \tag{5.2.11}$$

Die beiden Verschiebungsgleichungen (5.2.9) und (5.2.11) für den Biegestab nach der Theorie von TIMOSHENKO sind einseitig gekoppelt. Um $\varphi(x)$ aus Gl. (5.2.11) zu eliminieren, muss Gl. (5.2.11) dreimal differenziert werden:

$$w^{IV} = \frac{EI}{GA_s}\varphi^{V} - \varphi'''$$

Gl.(5.2.9) einsetzen: $\boxed{w^{IV} = -\frac{1}{GA_s}q''(x) + \frac{1}{EI}q(x)}$

5.2.3 Herleitung des P.v.V. für den schubweichen Biegestab (Variationsgleichungen)

Zur Aufstellung der Variationsgleichung werden die GGW-Bedingungen

$$N' + n = 0$$
$$Q' + q = 0$$
$$\text{und } M' - Q = 0$$

mit den Wichtungsfunktionen δu, δw und $\delta\varphi$ multipliziert, die Produkte addiert und über das Gebiet integriert, liefert das Prinzip der virtuellen Verschiebung (P.v.V.) für den Balken nach der TIMOSHENKO-Theorie:

$$\int_0^l \left[\delta u\left(N'+n\right) + \delta w\left(Q'+q\right) + \delta\varphi\left(M'-Q\right)\right] \mathrm{d}x = 0$$

Elastizitätsgleichungen und kinematische Gleichungen einsetzen:

$$-\int_0^l \left\{\delta u\left[EA\,u'\right]' + \delta w\left[GA_s\left(w'+\varphi\right)\right]' + \delta\varphi\left[\left(EI\,\varphi'\right)' - GA_s\left(w'+\varphi\right)\right]\right\} \mathrm{d}x = \int_0^l \left(\delta u\, n + \delta w\, q\right) \mathrm{d}x$$

Die partielle Integration des P.v.V. mit den Elastizitätsgleichungen liefert die schwache Form der Gleichgewichtsaussage für den Balken nach TIMOSHENKO:

$$\int_0^l \left[\delta u'\, EA u' + \delta w'\, GA_s \left(w' + \varphi \right) + \delta\varphi' \left(EI\varphi' \right) + \delta\varphi\, GA_s \left(w' + \varphi \right) \right] \mathrm{d}x = \int_0^l \left(\delta u\, n + \delta w\, q \right) \mathrm{d}x +$$

$$+ \left[\delta u \left(EAu' \right) \right]_{x=0}^{l} + \left[\delta w \left(GA_s \left(w' + \varphi \right) \right) \right]_0^l + \left[\delta\varphi \left(EI\varphi' \right) \right]_0^l$$

Die Wichtungsfunktionen δu, δw und $\delta\varphi$ sollen die kinematischen Randbedingungen am Verschiebungsrand $x = x_u$ erfüllen, d.h.

$$\delta u\big|_{x=x_{u_x}} = 0$$

$$\delta w\big|_{x=x_{u_z}} = 0$$

$$\delta w'\big|_{x=x_{u_\varphi}} = 0$$

Die Wichtungsfunktionen entsprechen dann den virtuellen Verschiebungen.

Am Kräfterand bei $x = x_{Px}$ gilt: $N\left(x = x_{px} \right) = P_x$

$x = x_{Pz}$ $\quad Q\left(x = x_{pz} \right) = P_z$

$x = x_{Me}$ $\quad M\left(x = x_{Me} \right) = M_e$

Der kinematische Operator **L** wird zwischen den Verschiebungen bzw. Verdrehung $\mathbf{u} = \left[u\ w\ \varphi \right]^{\mathrm{T}}$ und den Verzerrungen bzw. der Verkrümmung $\left[\varepsilon\ \gamma\ \kappa \right]^{\mathrm{T}}$ definiert:

$$\begin{bmatrix} \varepsilon \\ \gamma \\ \kappa \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{d}{dx} & 0 & 0 \\ 0 & \frac{d}{dx} & 1 \\ 0 & 0 & \frac{d}{dx} \end{bmatrix}}_{\mathbf{L}} \begin{bmatrix} u \\ w \\ \varphi \end{bmatrix} = \mathbf{Lu}$$

Einsetzen des kinematischen Operators **L** in die schwache Form liefert die Variationsgleichung für den Biegestab nach der TIMOSHENKO-Theorie:

$$\int_0^l \left[\delta u\ \ \delta w\ \ \delta\varphi \right] \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 0 & 0 \\ 0 & \frac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 0 & 1 & \frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} \begin{bmatrix} EA & 0 & 0 \\ 0 & GA_s & 0 \\ 0 & 0 & EI \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 0 & 0 \\ 0 & \frac{\mathrm{d}}{\mathrm{d}x} & 1 \\ 0 & 0 & \frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} \begin{bmatrix} u \\ w \\ \varphi \end{bmatrix} \mathrm{d}x =$$

$$= \left[\delta u\, P_x \right]_{x=xP_x} + \left[\delta w\, P_z \right]_{x=xP_z} + \left[\delta\varphi\, M_e \right]_{x=x_{Me}} + \int_0^l \left[\delta u\ \ \delta w \right] \begin{bmatrix} n \\ q \end{bmatrix} \mathrm{d}x$$

Zwischenrechnung zur Aufspaltung der virtuellen Änderung der Formänderungsenergie des Biegestabs mit Querschubverformung in einen reinen Steifigkeitsanteil für die Querschubdeformationen und in jenen für die Krümmung (reine Biegung):

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 1 & \frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} \begin{bmatrix} GA_s & 0 \\ 0 & EI \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 1 \\ 0 & \frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} = \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x}GA_s & 0 \\ GA_s & \frac{\mathrm{d}}{\mathrm{d}x}EI \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 1 \\ 0 & \frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} = \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x}GA_s\frac{\mathrm{d}}{\mathrm{d}x} & GA_s\frac{\mathrm{d}}{\mathrm{d}x} \\ \left(GA_s\right)\frac{\mathrm{d}}{\mathrm{d}x} & GA_s + \frac{\mathrm{d}}{\mathrm{d}x}EI\frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x}GA_s\frac{\mathrm{d}}{\mathrm{d}x} & GA_s\frac{\mathrm{d}}{\mathrm{d}x} \\ \frac{\mathrm{d}}{\mathrm{d}x}GA_s & GA_s + \left(\frac{\mathrm{d}}{\mathrm{d}x}\right)EI\frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} = \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} GA_s & GA_s \\ GA_s & GA_s \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 0 & \frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & EI \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 0 & \frac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix}$$

Aufspaltung der Steifigkeiten in die beiden Anteile:

Schub-steifigkeit

Biege-steifigkeit

5.2.4 Steifigkeitsmatrix mit linearen Ansätzen für das schubweiche Balkenelement nach der TIMOSHENKO-Theorie

Die Elementfreiheitsgrade des Biegestabs und die betreffende Einheitskoordinate ξ mit $-1 \le \xi \le 1$ werden in Abb. 5.2-1 dargestellt.

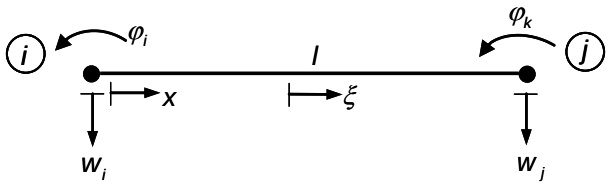


Abb. 5.2-1: Freiheitsgerade des Biegestabs nach TIMOSHENKO

$$x = \frac{l}{2}(1+\xi)$$

$$\mathrm{d}x = \frac{l}{2}\mathrm{d}\xi$$

Lineare Formfunktionen werden für die Diskretisierung (Interpolation) des Verschiebugnsfeldes $w_h(x)$ und der Querschnittsverdrehung $\varphi_h(x)$ angenommen. Sie sind in Abb. 5.2-2 als Einheitsverschiebungszustände abgebildet.

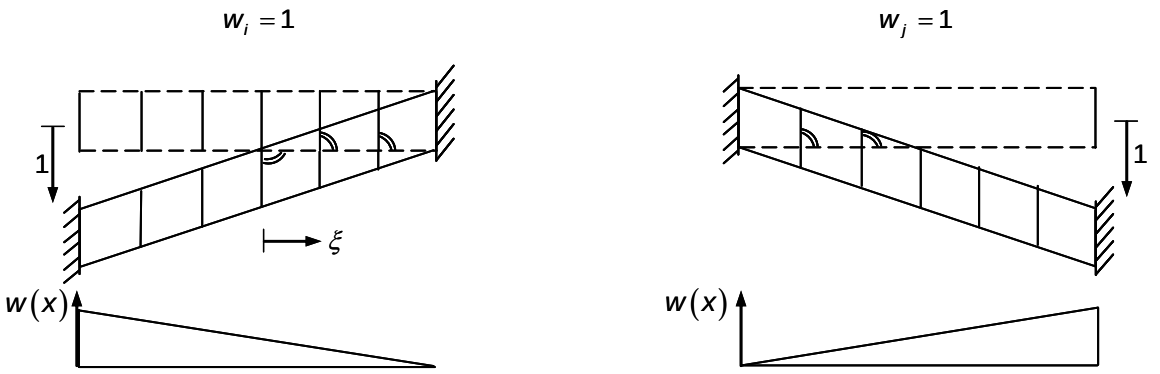


Abb. 5.2-2: Formfunktionen $w_i = 1$ und $w_j = 1$ für die Querverschiebung

Keine Querschnittsverdrehung $\varphi(x)$

$$N_2 = \frac{1}{2}(1-\xi) \qquad N_5 = \frac{1}{2}(1+\xi); \quad \frac{\mathrm{d}N_5}{\mathrm{d}x} = +\frac{1}{l}$$

$$\frac{\mathrm{d}N_2}{\mathrm{d}x} = -\frac{1}{l} \qquad \frac{\mathrm{d}N_5}{\mathrm{d}x} = +\frac{1}{l}$$

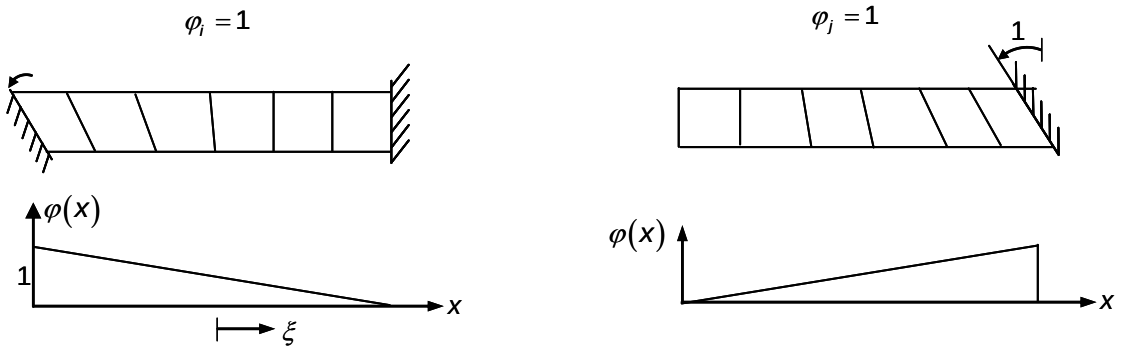


Abb. 5.2-3: Formfunktionen $\varphi_i = 1$ und $\varphi_j = 1$ für die Querverschiebung

Keine Querschnittsverschiebung $w(x)$

$$N_3 = \frac{1}{2}(1-\xi) \qquad N_6 = \frac{1}{2}(1+\xi)$$

$$\frac{\mathrm{d}N_3}{\mathrm{d}x} = -\frac{1}{l} \qquad \frac{\mathrm{d}N_6}{\mathrm{d}x} = +\frac{1}{l}$$

Interpolation des Verschiebungsfeldes:

$$w_{\mathrm{h}}(\xi) = \begin{bmatrix} N_2 & 0 & N_5 & 0 \end{bmatrix} \begin{bmatrix} w_i \\ \varphi_i \\ w_j \\ \varphi_j \end{bmatrix} = \mathbf{N}_w \ \mathbf{u}_{\mathrm{B}}$$

Interpolation der Querschnittsverdrehung:

$$\varphi_{\mathrm{h}}(\xi) = \begin{bmatrix} 0 & N_3 & 0 & N_6 \end{bmatrix} \begin{bmatrix} w_i \\ \varphi_i \\ w_j \\ \varphi_j \end{bmatrix} = \mathbf{N}_{\varphi}\ \mathbf{u}_{\mathrm{B}}$$

Die Krümmung ergibt sich aus dem Verlauf der Querschnittsverdrehung $\varphi_{\mathrm{h}}(x)$ zu: $\kappa \cong \kappa_{\mathrm{h}} = \varphi'_{\mathrm{h}} = \mathbf{N}'_{\varphi}\ \mathbf{u}_{\mathrm{B}}$

$$\kappa_{\mathrm{h}} = \begin{bmatrix} 0 & -\frac{1}{l} & 0 & \frac{1}{l} \end{bmatrix} \begin{bmatrix} w_i \\ \varphi_i \\ w_j \\ \varphi_j \end{bmatrix}$$

Der Krümmungsverlauf $\kappa_{\mathrm{h}}(x)$ aus der diskretisierten Querschnittsverdrehung $\varphi_{\mathrm{h}}(x)$ ist konstant über die Stablänge – siehe Abb. 5.2-4

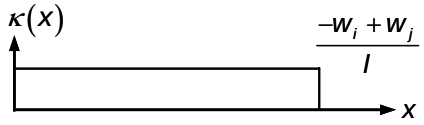


Abb.: 5.2-4: Konstanter Krümmungsverlauf

Die Querschubverzerrung folgt für den Biegestab mit obiger Diskretisierung $w_{\mathrm{h}}(x)$ und $\varphi_{\mathrm{h}}(x)$ zu:

$$\gamma_{xz_{\mathrm{h}}} = w'_{\mathrm{h}} + \varphi_{\mathrm{h}} = \mathbf{N}'_{w}\ \mathbf{u}_{\mathrm{B}} + \mathbf{N}_{\varphi}\ \mathbf{u}_{\mathrm{B}} = \left[\mathbf{N}'_{w} + \mathbf{N}_{\varphi}\right] \mathbf{u}_{\mathrm{B}}$$

$$\gamma_{xz_{\mathrm{h}}} = \begin{bmatrix} -\frac{1}{l} & \frac{1}{2}(1-\xi) & \frac{1}{l} & \frac{1}{2}(1+\xi) \end{bmatrix} \begin{bmatrix} w_i \\ \varphi_i \\ w_j \\ \varphi_j \end{bmatrix}$$

Die Querschubverzerrung $\gamma_{xz_{\mathrm{h}}}$ verläuft im diskretisierten Biegestab linear über die Balkenlänge.

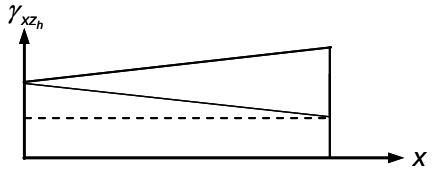


Abb. 5.2-5: Linearer Verlauf der Querschubverzerrung

Die voneinander unabhängigen Feldvariablen für die Querverschiebung w_{h} und die Querschnittsverdrehung φ_{h} werden zum Spaltenvektor **u** zusammengefasst.

$$\mathbf{u} = \begin{bmatrix} w_{\mathrm{h}} \\ \varphi_{\mathrm{h}} \end{bmatrix} = \begin{bmatrix} \mathbf{N}_{w} \\ \mathbf{N}_{y} \end{bmatrix} \mathbf{u}_{\mathrm{B}} = \mathbf{N}\mathbf{u}_{\mathrm{B}}$$

Die kinematischen Operatoren für die Biegung $\mathbf{L}_\mathrm{B}$ und die Querschubverzerrung $\mathbf{L}_\mathrm{S}$ werden auf den Vektor der Formfunktionen $\mathbf{N}$ angewendet:

$$\mathbf{B}_\mathrm{B} = \mathbf{L}_\mathrm{B}\,\mathbf{N} = \begin{bmatrix} \dfrac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{N}_w \\ \mathbf{N}_\varphi \end{bmatrix} = \begin{bmatrix} \mathbf{N}'_w \\ \mathbf{N}_\varphi \end{bmatrix}$$

$$\mathbf{B}_\mathrm{S} = \mathbf{L}_\mathrm{S}\,\mathbf{N} = \begin{bmatrix} \dfrac{\mathrm{d}}{\mathrm{d}x} & 0 \\ 0 & \dfrac{\mathrm{d}}{\mathrm{d}x} \end{bmatrix} \begin{bmatrix} \mathbf{N}_w \\ \mathbf{N}_\varphi \end{bmatrix} = \begin{bmatrix} \mathbf{N}'_w \\ \mathbf{N}_\varphi \end{bmatrix}$$

Elementbiegesteifigkeit:

$$\mathbf{k}_\mathrm{B} = \int_0^l \begin{bmatrix} \mathbf{N}'^{\mathrm{T}}_w & \mathbf{N}'^{\mathrm{T}}_\varphi \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & EI \end{bmatrix} \begin{bmatrix} \mathbf{N}'_w \\ \mathbf{N}'_\varphi \end{bmatrix} \mathrm{d}x = \int_0^l \mathbf{N}'^{\mathrm{T}}_\varphi\, EI\, \mathbf{N}'^{\mathrm{T}}_\varphi\, \mathrm{d}x$$

$$\int_{-1}^{+1} \begin{bmatrix} 0 \\ -\dfrac{1}{l} \\ 0 \\ \dfrac{1}{l} \end{bmatrix} EI \begin{bmatrix} 0 & -\dfrac{1}{l} & 0 & \dfrac{1}{l} \end{bmatrix} \frac{l}{2}\,\mathrm{d}\xi = \frac{EI}{l} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

Elementschubsteifigkeit:

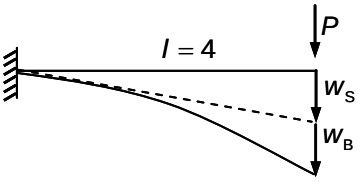
$$\mathbf{k}_\mathrm{S} = \int_0^l \begin{bmatrix} \mathbf{N}'^{\mathrm{T}}_w & \mathbf{N}^{\mathrm{T}}_\varphi \end{bmatrix} \begin{bmatrix} GA_\mathrm{s} & GA_\mathrm{s} \\ GA_\mathrm{s} & GA_\mathrm{s} \end{bmatrix} \begin{bmatrix} \mathbf{N}'_w \\ \mathbf{N}_\varphi \end{bmatrix} \mathrm{d}x$$

$$\mathbf{k}_\mathrm{s} = \int_{-1}^{+1} \begin{bmatrix} -\dfrac{1}{l} & 0 \\ 0 & \dfrac{1}{l}(1-\xi) \\ +\dfrac{1}{l} & 0 \\ 0 & \dfrac{1}{2}(1+\xi) \end{bmatrix} \begin{bmatrix} GA_\mathrm{s} & GA_\mathrm{s} \\ GA_\mathrm{s} & GA_\mathrm{s} \end{bmatrix} \begin{bmatrix} -\dfrac{1}{l} & 0 & \dfrac{1}{l} & 0 \\ 0 & \dfrac{1}{2}(1-\xi) & 0 & \dfrac{1}{2}(1+\xi) \end{bmatrix} \frac{l}{2}\,\mathrm{d}\xi$$

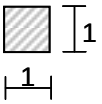
$$\mathbf{k}_\mathrm{s} = \int_{-1}^{+1} GA_\mathrm{s} \begin{bmatrix} \dfrac{1}{l^2} & -\dfrac{1}{2l}(1-\xi) & -\dfrac{1}{l^2} & -\dfrac{1}{2l}(1+\xi) \\ -\dfrac{1}{2l}(1-\xi) & \dfrac{1}{4}(1-\xi)^2 & \dfrac{1}{2l}(1-\xi) & \dfrac{1}{4}(1-\xi)^2 \\ -\dfrac{1}{l^2} & \dfrac{1}{2l}(1-\xi) & \dfrac{1}{l^2} & \dfrac{1}{2l}(1+\xi) \\ -\dfrac{1}{2l}(1+\xi) & \dfrac{1}{4}(1-\xi) & \dfrac{1}{2}(1+\xi) & \dfrac{1}{4}(1+\xi)^2 \end{bmatrix} \frac{l}{2}\,\mathrm{d}\xi$$

$$\mathbf{k}_S = \frac{GA_s}{l}\begin{bmatrix} 1 & -\frac{l}{2} & -1 & -\frac{l}{2} \\ -\frac{l}{2} & \frac{l^2}{3} & \frac{l}{2} & \frac{l^2}{6} \\ -1 & \frac{l}{2} & 1 & \frac{l}{2} \\ -\frac{l}{2} & \frac{l^2}{6} & \frac{l}{2} & \frac{l^2}{3} \end{bmatrix}$$

Beispiel: FE-Berechnung eines Kragarms unter Querlast



Querschnitt



$$A_s = \frac{5}{6}1\cdot 1 = \frac{5}{6}$$

$$I = \frac{1}{12}1\cdot 1^3 = \frac{1}{12}$$

Durchbiegung:

$$w^{\text{anal}} = w_B + w_S$$

$$= \frac{1}{3}\frac{Pl^3}{EI} + \frac{Pl}{GA_S}$$

$$= \frac{Pl^3}{3EI}\left(1 + \frac{3EI}{GA_S l^2}\right)$$

Elastizitätskennwert: $E = 10^3$

schubweich: $G = 375$

"schubstarr": $G = 375\cdot 10^5$

Konvergenzstudie für den TIMOSHENKO-Balken mit linearen Ansatzfunktionen liefert für den schubweichen $(G = 375)$ und „schubstarren“ $(G = 375\cdot 10^5)$ Kragarm im ersten Fall brauchbare und im zweiten Fall viel zu kleine, unbrauchbare Ergebnisse. Letzteres Verhalten wird als „shear locking“ (exzessive Schubversteifung) bezeichnet.

# Elemente	normierte Verschiebung schubweicher Balken	normierte Verschiebung schubstarrer Balken
1	0,042	0,00002
2	0,445	0,00008
4	0,762	0,00012
8	0,927	0,00032
16	0,981	0,00051

$$\text{normierte Verschiebung} = \frac{w^{\text{numerisch}}}{w^{\text{analytisch}}}$$

Die Integrale zur Berechnung der Komponenten der Steifigkeitsmatrix können durch die numerische Integration nach GAUß berechnet werden. Wegen der quadratischen Terme ξ^2 ist dafür die Formel mit zwei Stützstellen nach GAUß notwendig.

Die exzessive Versteifung („shear locking“) verschwindet, wenn die GAUß-Quadratur mit einer Stützstelle verwendet wird. In diesem Fall werden quadratische und höhere Polynomterme ξ^n für $n \geq 2$ durch das numerische Integrationsverfahren nicht mehr erfasst.

Dieses Verfahren ist gleichbedeutend, wenn die Terme mit ξ^2 in der Steifigkeitsmatrix $\mathbf{k}_s$ gestrichen werden.

$$\mathbf{k}_S^{\text{red}} = \int_{-1}^{+1} GA_s \begin{bmatrix} \frac{1}{l^2} & -\frac{1}{2l}(1-\xi) & -\frac{1}{l^2} & -\frac{1}{2l}(1+\xi) \\ -\frac{1}{2l}(1-\xi) & \frac{1}{4} & \frac{1}{2l}(1-\xi) & \frac{1}{4} \\ -\frac{1}{l^2} & \frac{1}{2l}(1-\xi) & \frac{1}{l^2} & \frac{1}{2l}(1+\xi) \\ -\frac{1}{2l}(1+\xi) & \frac{1}{2l} & \frac{1}{2l}(1+\xi) & \frac{1}{4} \end{bmatrix} \frac{l}{2}\,\mathrm{d}\xi$$

$$\mathbf{k}_S^{red} = \frac{GA_s}{l}\begin{bmatrix} 1 & -\frac{l}{2} & -1 & -\frac{l}{2} \\ -\frac{l}{2} & \frac{l^2}{4} & \frac{l}{2} & \frac{l^2}{4} \\ -1 & \frac{l}{2} & 1 & \frac{l}{2} \\ -\frac{l}{2} & \frac{l^2}{4} & \frac{l}{2} & \frac{l^2}{4} \end{bmatrix}$$

Die Biegesteifigkeit $\mathbf{k}_B$ bleibt unverändert.

Die reduziert integrierte Schubsteifigkeitsmatrix $\mathbf{k}^{red}$ ergibt für ein grobes Netz mit wenig Elementen beim schubweichen Balken deutlich bessere Verschiebungen.

Die Ergebnisse für den schubstarren Kragarm liegen für ein Element in der Nähe der exakten Lösung. Bei Elementverfeinerung konvergiert die Approximation ähnlich gut gegen die exakte Lösung wie jene für den schubweichen Balken.

# Elemente	normierte Verschiebung schubweicher Balken	normierte Verschiebung schubstarrer Balken
1	0,762	0,750
2	0,940	0,978
4	0,985	0,984
8	0,996	0,996
16	0,999	0,999

Fazit: Behebung des „locking“-Effekts mittels reduzierter Integration der Schubsteifigkeit $\mathbf{k}_S$ mit einem einzelnen GAUß-Punkt.

5.2.5 Diskretisierung der Felder für die Längs- und Querverschiebung sowie Querschnittsverdrehung

Die Verschiebungsfelder werden mit Formfunktionen und Knotenverschiebungen diskretisiert.

5.2.5.1 Approximation mittels der homogenen exakten Lösung der Verschiebungsgleichungen

Die Formfunktionen für den schubweichen Biegestab werden aus den Lösungen der Verschiebungsgleichungen für die Einheitsverschiebungszustände gewonnen.

Die homogene DGL – siehe Gl. (4.6.6) mit **q** (**x**) =0 :

$$\left[EI\,\varphi'\right]'' = 0$$

liefert die Lösung für die Verdrehung $\varphi(x)$ des Querschnitts:

$$\varphi(x) = c_1 + c_2 x + c_3 x^2$$

Die Querverschiebung $w(x)$ folgt aus Gl. (4.6.8)

$$w' = \frac{EI}{GA_s} 2c_3 - c_1 - c_2 x - c_3 x^2$$

zu:

$$w(x) = -\frac{1}{3}c_3\, x^3 - \frac{1}{2}c_2\, x^2 - \left(c_1 - 2c_3 \frac{EI}{GA_s}\right)x + c_0$$

Anpassung der allgemeinen Lösungen φ (**x**) und **w** (**x**) an die Randbedingungen:

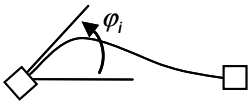
$$w(x=0) = w_i = 1, \quad \varphi(x=0) = 0, \quad w(x=l) = 0, \quad \varphi(x=l) = 0$$

$$c_0 = 1$$

$$c_1 = 0$$

$$c_2 = \frac{6GA_s}{GA_s l^2 + 12\,EI}$$

$$c_3 = -\frac{6GA_s}{GA_s l^3 + 12\,EI\,l}$$



$$w(x=0) = 0, \quad \varphi(x=0) = 1, \quad w(x=l) = 0, \quad \varphi(x=l) = 0$$

$$c_0 = 0$$

$$c_1 = 1$$

$$c_2 = -\frac{4GA_s l^2 + 12\,EI}{GA_s l^3 + 12\,EI\,l}$$

$$c_3 = \frac{3GA_s}{GA_s l^2 + 12\,EI\,l}$$

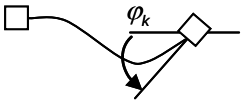
$$w(x=0)=0,\quad \varphi(x=0)=0,\quad w(x=l)=1,\quad \varphi(x=l)=0$$

$$c_0 = 0$$

$$c_1 = 0$$

$$c_2 = -\frac{6\,GA_s}{GA_s l^2 + 12\,EI}$$

$$c_3 = \frac{6GA_s}{GA_s l^3 + 12\,EI\,l}$$



$$w(x=0)=0,\quad \varphi(x=0)=0,\quad w(x=l)=0,\quad \varphi(x=l)=1$$

$$c_0 = 0$$

$$c_1 = 0$$

$$c_2 = -\frac{2\,GA_s\,l^2 - 12EI}{GA_s l^3 + 12\,EI\,l}$$

$$c_3 = \frac{3GA_s}{GA_s l^2 + 12\,EI}$$

5.2.5.2 Elementsteifigkeitsmatrix des TIMOSHENKO-Balkens

$$\mathbf{k} = \int_0^l \mathbf{B}^T \mathbf{C}\,\mathbf{B}\,dx$$

5.2.6 Steifigkeitsmatrix für das schubweiche Balkenelement nach TIMOSHENKO – ingenieuranschauliche Herleitung –

5.2.6.1 Einheitskraftzustände des Balkens mit Schubverzerrungen und Flexibilitätsmatrix

Schubverzerrungen können in den Balken auf zwei Arten erfasst werden:

- Ergänzung der EULER-BERNOULLI Balkentheorie um Schubverzerrungsanteile,
- Herleitung aus einer Balkentheorie mit Berücksichtigung der Schubverzerrung (TIMOSHENKO-Balken).

Hier wird der erste Weg eingeschlagen und anstelle der Herleitung über das P.v.V. ein anschauliches Vorgehen über die Flexibilitätsmatix **f** gewählt. Dabei können die Verformungen aus Biege- und Schubanteil addiert werden. Die Dehnanteile lassen sich wie üblich ergänzen.

a) Reiner Biegeanteil:

Flexibilitätsbeziehungen

$$\begin{bmatrix} \vartheta_i \\ \vartheta_k \end{bmatrix} = \frac{1}{6EI} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} M_i \\ M_k \end{bmatrix}$$

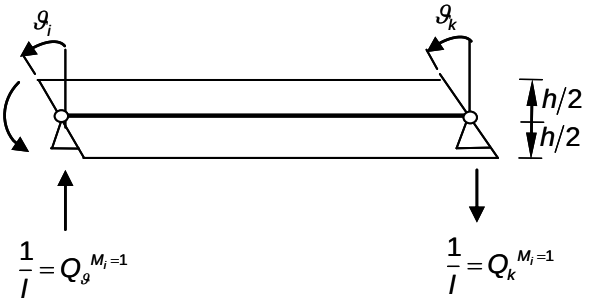
$\mathbf{f}_{\text{Biegung}}$

ϑ : Sehnenwinkel

Abb. 5.2-5: Balken unter reinen Biegeverformungen

b) Reiner Schubanteil:

$\boxed{M_i = 1}$



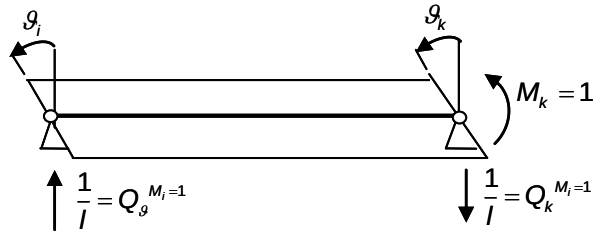
linearer Momentenzustand
konstanter Querkraftzustand

$$y_{xz} = \vartheta = \frac{Q}{G\alpha_Q A} =$$

$$= \frac{1}{G\alpha_Q Al} = \text{konstant}$$

Abb. 5.2-6: Balken unter Schubbeanspruchung infolge Einheitsmoment $M_i = 1$

$\boxed{M_k = 1}$



Flexibilitätsbeziehungen

$$\begin{bmatrix} \vartheta_i \\ \vartheta_k \end{bmatrix}_s = \frac{1}{G\alpha_Q Al} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} M_i \\ M_k \end{bmatrix}$$

$\mathbf{f}_{\text{Schub}}$

Abb. 5.2-7: Balken unter Schubbeanspruchung infolge Einheitsmoment $M_k = 1$

c) Superposition: Biege- und Schubanteil

Verformungen addieren sich

$$\mathbf{f} = \mathbf{f}_{\text{Biegung}} + \mathbf{f}_{\text{Schub}} = \frac{1}{6EI} \begin{bmatrix} 2+\frac{\eta}{2} & -1+\frac{\eta}{2} \\ \text{symm.} & 2+\frac{\eta}{2} \end{bmatrix}$$

$$\eta = \frac{12EI}{G \cdot \alpha_Q \cdot A \cdot l^2}$$

5.2.6.2 Überführung in Steifigkeitsmatrix bezüglich reiner Querschnittsverdrehungen relativ zur Stabsehne:

$$\mathbf{f}^{-1} = \mathbf{k}_{red} = \frac{2EI}{l(1+\eta)} \begin{bmatrix} 2+\frac{\eta}{2} & 1-\frac{\eta}{2} \\ \text{symm.} & 2+\frac{\eta}{2} \end{bmatrix}$$

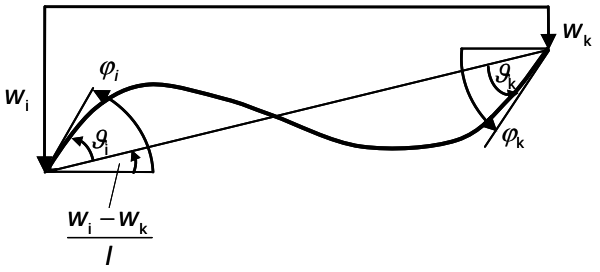
ohne Starrkörperverschiebungen

5.2.6.3 Erweiterung durch Starrkörperanteile

Gesamter Stabenddrehwinkel:

$$\varphi_{i\,k} = \vartheta_{i\,k} + \frac{w_i - w_k}{1}$$

mit Starrkörperverschiebung

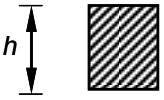


$$\begin{bmatrix} \vartheta_i \\ \vartheta_k \end{bmatrix} = \begin{bmatrix} -\frac{1}{l} & 1 & \frac{1}{l} & 0 \\ -\frac{1}{l} & 0 & \frac{1}{l} & 1 \end{bmatrix} \begin{bmatrix} w_i \\ \varphi_i \\ w_k \\ \varphi_k \end{bmatrix}$$

Abb. 5.2-8: Balken mit zusätzlichen Starrkörperverschiebungen

Transformation auf nicht reduzierte Steifigkeitsmatrix:

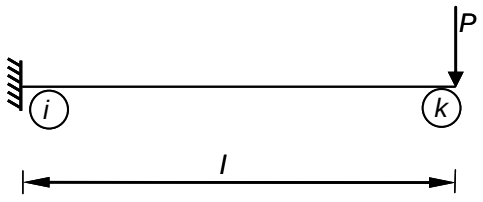
$$\mathbf{k} = \mathbf{\Gamma}^T \cdot \mathbf{k}_{red} \cdot \mathbf{\Gamma} = \frac{EI}{l^3(1+\eta)} \begin{bmatrix} 12 & -6l & -12 & -6l \\ & (4+\eta) & 6l & (2-\eta)l^2 \\ & & 12 & 6l \\ & & & (4+\eta)l^2 \end{bmatrix}$$

Rechteckquerschnitt $(\mu = 0)$: 

$$\eta = 2{,}4 \cdot \left(\frac{h}{l}\right)^2$$

Der Grenzfall $\eta = 0$ führt auf die Steifigkeitsmatrix ohne Schubverzerrungen.

Anmerkung



$$\frac{EI}{l^3(1+\eta)}\begin{bmatrix}12 & 6l\\ 6l & (4+\beta\eta)l^2\end{bmatrix}\begin{bmatrix}w_k\\ \varphi_k\end{bmatrix}=\begin{bmatrix}P\\ 0\end{bmatrix}$$

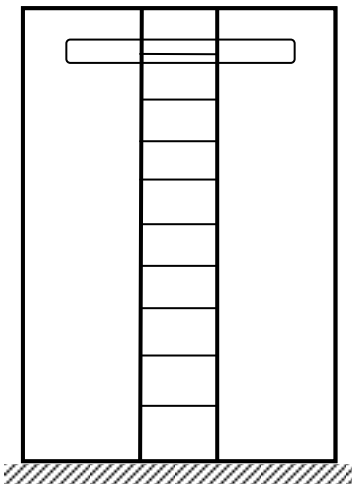
$$\left[12-\frac{36}{4+\eta}\right]\cdot w_k = \frac{P\cdot l^3}{EI}\cdot(1+\eta)$$

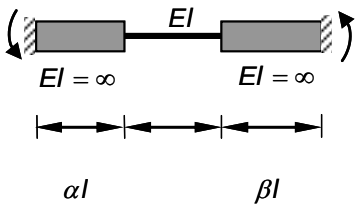
$$w_k = \frac{p\cdot l^3}{EI}\cdot\frac{4+\eta}{12}$$

Abb. 5.2-9*: Kragträger unter Einzellast mit Schubverformung

5.2.7 Balken mit starren Enden

Ohne weitere Herleitung wird die reduzierte Steifigkeitsmatrix für die Berechnung von Scheiben – Rahmen – Systemen mit starren Balkenenden angegeben. Wie im Abschnitt 4.7.1 kann die Matrix um die Starrkörperanteile erweitert werden.





$$\mathbf{k}_{red}=\frac{2EI}{l}\begin{bmatrix}1+\alpha & \alpha\\ \beta & 1+\beta\end{bmatrix}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}\begin{bmatrix}1+\alpha & \beta\\ \alpha & 1+\beta\end{bmatrix}$$

Abb. 5.2-10: Idealisierung mit Makroelementen

6 Platten

6.1 Definition der Schnittgrößen

Ebene, dünne Körper bezeichnet man als Platten oder Scheiben, je nachdem ob die Belastung quer oder tangential zur Mittelebene wirkt. (Die Definition der Schnittgrößen (Biegemomente und Querkräfte) erfolgt wie in der Vorlesung SPS)

Biegemomente:

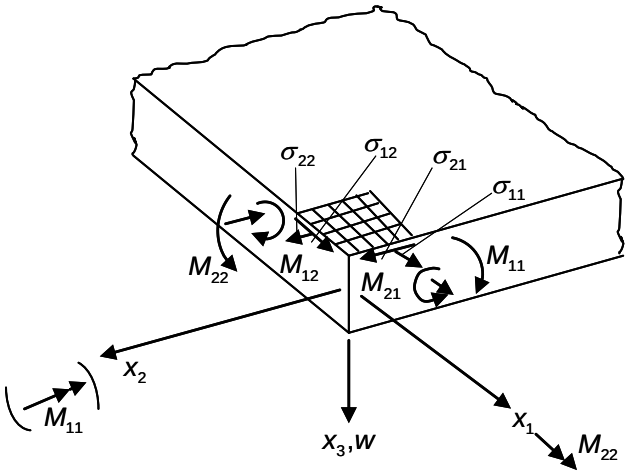


Abb. 6.1-1: Definition der Schnittmomente der Platte

Definition der Biegemomente: $M_{\alpha\beta} = -\int_{-h/2}^{h/2} x_3 \sigma_{\alpha\beta}\, dx_3$ für $\alpha = \beta$ und der Drillmomente für $\alpha \neq \beta$.

Querkräfte:

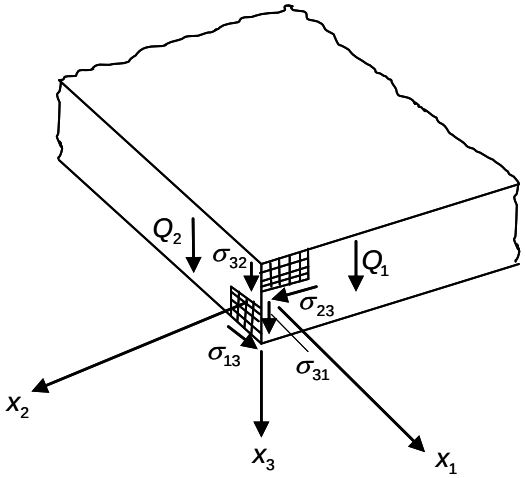


Abb. 6.1-2: Definition der Querkräfte der Platte

Definition der Querkräfte:

$$Q_\alpha = \int_{-h/2}^{h/2} \sigma_{3\alpha}\, dx_3$$

Nach dem Reaktionsprinzip sind die Schnittkraftresultierenden (Schnittgrößen) an gegenüberliegenden Schnittufern dem Betrag nach gleich aber im Vorzeichen unterschiedlich.

Für das Plattenelement gelten demnach folgende Definitionen:

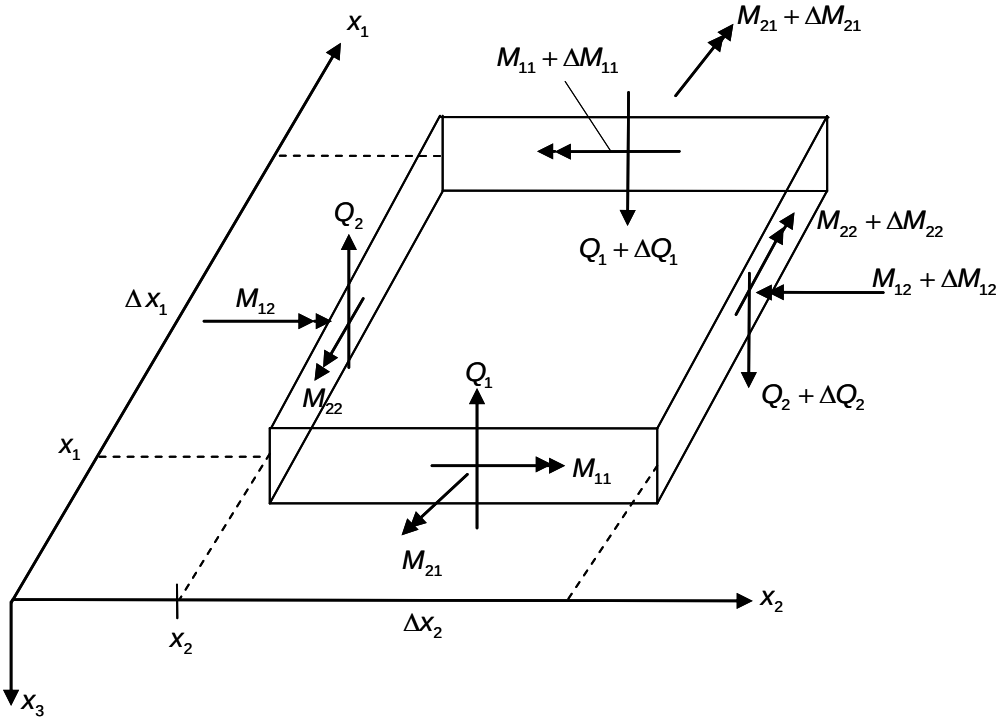


Abb. 6.1-3: Infinitesimales Plattenelement

6.2 Gleichgewichtsbedingungen

Mit den Methoden der Ingenieuranschauung werden die Gleichgewichtsbedingungen am Plattenelement $\Delta x_1, \Delta x_2$ aufgestellt.

$$\xrightarrow{-x_2} \sum M_{i\ s} = 0 = (M_{11}+\Delta M_{11})\Delta x_2 - M_{11}\cdot\Delta x_2 + (M_{12}+\Delta M_{12})\cdot\Delta x_1 - M_{12}\cdot\Delta x_1 +$$
$$+ (Q_1+\Delta Q_1)\Delta x_2 \frac{\Delta x_1}{2} + Q_1\cdot\Delta x_2 \frac{\Delta x_1}{2} \qquad \Big| : \Delta x_1\cdot\Delta x_2$$
$$\frac{\Delta M_{11}}{\Delta x_1} + \frac{\Delta M_{12}}{\Delta x_2} + Q_1 + \frac{\Delta Q_1}{2} = 0$$

$\Delta x_1 \to 0;\ \Delta x_2 \to 0$

$$\frac{\partial M_{11}}{\partial x_1} + \frac{\partial M_{12}}{\partial x_2} + Q_1 = 0$$

$$\overset{x_1}{\square}\quad \sum M_{is} = 0 = (M_{22}+\Delta M_{22})\Delta x_1 - M_{22}\Delta x_1 + (M_{21}+\Delta M_{21})\Delta x_2 - M_{21}\Delta x_2 + (Q_2\Delta x_1)\frac{\Delta x_2}{2} + (Q_2+\Delta Q_2)\Delta x_1\frac{\Delta x_2}{2}$$
$$\frac{\Delta M_{21}}{\Delta x_1} + \frac{\Delta M_{22}}{\Delta x_2} + Q_2 + \frac{\Delta Q_2}{2} = 0$$

$\Delta x_1 \to 0;\ \Delta x_2 \to 0:$ $\boxed{\frac{\partial M_{21}}{\partial x_1} + \frac{\partial M_{22}}{\partial x_2} + Q_2 = 0}$

$$+\downarrow \sum F_{ix} = 0 = (Q_1 + \Delta Q_1)\cdot \Delta x_2 - Q_1 \cdot \Delta x_2 + (Q_2 + \Delta Q_2)\cdot \Delta x_1 - Q_2 \cdot \Delta x_2 + q(x_1, x_2)\cdot \Delta x_1 \cdot \Delta x_2 = 0$$

$$\frac{\Delta Q_1}{\Delta x_1} + \frac{\Delta Q_2}{\Delta x_2} + q = 0$$

$\Delta x_1 \to 0;\ \Delta x_2 \to 0:$

$$\boxed{\frac{\partial Q_1}{\partial x_1} + \frac{\partial Q_2}{\partial x_2} + q = 0}$$

Matrizenschreibweise:

$$\begin{bmatrix} \frac{\partial}{\partial x_1} & 0 & \frac{\partial}{\partial x_2} \\ 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{bmatrix} \begin{bmatrix} M_{11} \\ M_{22} \\ M_{12} \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = 0$$

wobei gilt: $M_{12} = M_{21}$ (wegen $\sigma_{12} = \sigma_{21}$ - Momentengleichgewicht)

$$\boxed{\mathbf{L}^T \quad \mathbf{M} \quad + \quad \mathbf{Q} \quad = \quad \mathbf{0}}$$

und $\mathbf{L}^T$ ist der statische Operator für das Momentengleichgewicht

$$\begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} + \mathbf{q} = \mathbf{0}$$

$$\boxed{\boldsymbol{\nabla}^{\mathbf{T}}\mathbf{Q} + \mathbf{q} = \mathbf{0}}$$

Hinweis:

- Gradient ∇ ist ein Zeilenvektor $\boldsymbol{\nabla} := \mathbf{e}_1 \frac{\partial}{\partial x_1} + \mathbf{e}_2 \frac{\partial}{\partial x_2} = \begin{pmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} \end{pmatrix}$

  Wird aber hier als Spaltenvektor geschrieben.

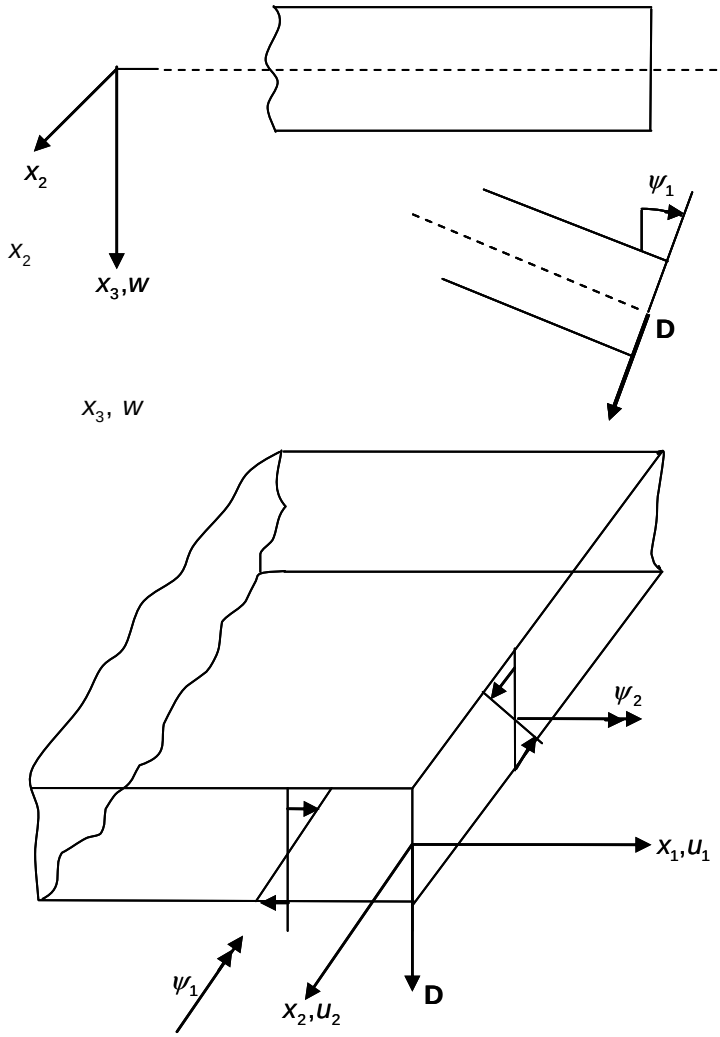
- Die Verknüpfung der Platten- mit der Scheibentheorie ergibt die Faltwerktheorie.

6.3 Kinematische Beziehungen

Querschnitte bleiben eben und verdrehen sich. Dabei verschiebt sich die Referenzfläche in x_3-Richtung.

Verschiebungen in der Plattenebene: u_1 (in x_1-Richtung) und u_2 in x_1-Richtung)

Verschiebungen quer zur Plattenebene: u_3 (in x_3-Richtung).



ψ_1 : Verdrehung um negative x_2 Achse

$$u_1(x_1, x_2, x_3) = -x_3 \cdot \psi_1(x_1, x_2)$$

D : Plattendirektor

ψ_2 : Verdrehung um $(+x_1)$ – Achse

$$u_2(x_1, x_2, x_3) = -x_3\ \psi_2(x_1, x_2)$$

Abb. 6.3-1: Kinematik der Platte

Direktor **D** ist dehnstarr, verdreht sich um die Schalenmittelfläche und verschiebt sich nur in x_3 – Richtung.

$$u_3(x_1, x_2, x_3 = 0) = w(x_1, x_2)$$

Dehnungen in der Plattenebene:

$$\varepsilon_{11} = \frac{\partial u_1}{\partial x_1} = -x_3 \frac{\partial \psi_1}{\partial x_1}$$

$$\varepsilon_{22} = \frac{\partial u_2}{\partial x_2} = -x_3 \frac{\partial \psi_2}{\partial x_2}$$

$$2\varepsilon_{12} = \gamma_{12} = \frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} = -x_3 \frac{\partial \psi_2}{\partial x_1} - x_3 \frac{\partial \psi_1}{\partial x_2}$$

Querschubverzerrungen:

$$2\varepsilon_{13} = \gamma_{13} = \frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3} = \frac{\partial w}{\partial x_1} - \psi_1$$

$$2\varepsilon_{23} = \gamma_{23} = \frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3} = \frac{\partial w}{\partial x_2} - \psi_2$$

In Matrizenschreibweise gilt für die kinematischen Beziehungen

- in der Plattenebene:

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{bmatrix} = -x_3 \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 \\ 0 & \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

$$\boxed{\boldsymbol{\varepsilon} = -x_3 \; \mathbf{L} \; \boldsymbol{\psi}}$$

wobei $\mathbf{L}$ der kinematische Operator ist.

Krümmungstensor:

$$\boldsymbol{\kappa} = \begin{bmatrix} \kappa_{11} & \kappa_{12} \\ \kappa_{21} & \kappa_{22} \end{bmatrix} = \begin{bmatrix} \psi_{1,1} & \psi_{2,1} + \psi_{1,2} \\ \psi_{1,2} + \psi_{2,1} & \psi_{2,2} \end{bmatrix}$$

$$\boldsymbol{\kappa} = \frac{1}{2}\left[\nabla\boldsymbol{\psi} + (\nabla\boldsymbol{\psi})^{\mathrm{T}}\right]$$

- quer zur Plattenebene:

$$\begin{bmatrix} \gamma_{13} \\ \gamma_{23} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \end{bmatrix} w - \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

$$\begin{bmatrix} \gamma_{13} \\ \gamma_{23} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} & -1 & 0 \\ \frac{\partial}{\partial x_2} & 0 & -1 \end{bmatrix} \begin{bmatrix} w \\ \psi_1 \\ \psi_2 \end{bmatrix}$$

$$\boxed{\boldsymbol{\gamma} = \nabla w - \boldsymbol{\psi}}$$

Für schubstarre Platten gilt: $\boldsymbol{\gamma} = \mathbf{0}$ und es ergibt sich die Plattentheorie mit der Zwangsbedingung: $\boldsymbol{\psi} = \nabla w$

(KIRCHHOFFsche Plattentheorie)

Die Dehnungen in der Plattenebene lauten dann:

$$\boldsymbol{\varepsilon} = -x_3 \; \mathbf{L}\nabla \mathrm{w} = -x_3 \boldsymbol{\kappa}$$

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{bmatrix} = -x_3 \begin{bmatrix} \dfrac{\partial^2}{\partial x_1^2} \\ \dfrac{\partial^2}{\partial x_2^2} \\ 2\,\dfrac{\partial^2}{\partial x_1 \partial x_2} \end{bmatrix} w \qquad \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{Krümmungen} \qquad \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{Verwindung (Drillung)}$$

6.4 Konstitutive Beziehungen

Elastizitätsgleichungen (in VOIGTscher Notation) – siehe TM I/II, S. 336

$$\varepsilon_{11} = \frac{1}{E}\left[\sigma_{11} - \nu\left(\sigma_{22} + \sigma_{33}\right)\right]$$

$$\varepsilon_{22} = \frac{1}{E}\left[\sigma_{22} - \nu\left(\sigma_{33} + \sigma_{11}\right)\right]$$

$$\varepsilon_{33} = \frac{1}{E}\left[\sigma_{33} - \nu\left(\sigma_{11} + \sigma_{22}\right)\right]$$

$$\gamma_{12} = \frac{1}{G}\,\sigma_{12}$$

$$\gamma_{13} = \frac{1}{G}\,\sigma_{31}$$

$$\gamma_{23} = \frac{1}{G}\,\sigma_{32} \qquad ; \qquad G = \frac{E}{2\left(1+\nu\right)}$$

Annahme: $\sigma_{33} = 0 \quad \Rightarrow \quad \varepsilon_{33} = -\dfrac{\nu}{E}\left(\sigma_{11} + \sigma_{22}\right)$

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \end{bmatrix} = \frac{1}{E}\begin{bmatrix} 1 & -\nu \\ -\nu & 1 \end{bmatrix}\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \end{bmatrix}$$

Invertieren:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix}\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{bmatrix} \qquad \textit{ESZ}$$

$$\begin{bmatrix} \sigma_{31} \\ \sigma_{32} \end{bmatrix} = \frac{E}{2\left(1+\nu\right)}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \gamma_{31} \\ \gamma_{32} \end{bmatrix}$$

Einsetzen in die Definitionsgleichungen für die Schnittgrößenresultierenden.

Für die Momente gilt:

$$\begin{bmatrix} M_{11} \\ M_{22} \\ M_{12} \end{bmatrix} = -\int_{-h/2}^{h/2} x_3 \, \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} (-x_3) \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 \\ 0 & \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} dx_3$$

$$\begin{bmatrix} M_{11} \\ M_{22} \\ M_{12} \end{bmatrix} = \frac{E\,h^3}{12\left(1-\nu^2\right)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 \\ 0 & \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

$$D = \frac{E\,h^3}{12\left(1-\nu^2\right)}$$

$$\mathbf{M} = \mathbf{D}\ \mathbf{L}\ \boldsymbol{\psi}$$

Elastizitätsgleichung der Plattenbiegung mit **D** als Matrix der Biege- und Drillsteifigkeiten der Platte.

Für den Querschub muss ein Schubkorrekturfaktor κ eingefügt werden, damit der parabolische Schubspannungsverlauf durch einen konstanten Schubspannungsverlauf ersetzt werden darf.

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \kappa \int_{-h/2}^{h/2} \frac{E}{2\left(1+\nu\right)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \left\{ \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \end{bmatrix} w - \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} \right\} dx_3$$

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \kappa \frac{E\,h}{2\left(1+\nu\right)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \left\{ \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \end{bmatrix} w - \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} \right\}$$

$$\mathbf{Q} = \boldsymbol{\alpha}\ \left(\nabla w - \boldsymbol{\psi}\right)$$

wobei $\boldsymbol{\alpha} = \kappa\, G h \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ die Schubsteifigkeit der schubweichen Plattentheorie nach REISSNER-MINDLIN ist.

$$\boldsymbol{\alpha} = \kappa G h \mathbf{I} = \alpha \mathbf{I} \quad \text{mit} \quad \alpha = \kappa G h$$

Für isotrope Elastizität folgt:

$$\mathbf{Q} = \alpha\, \mathbf{I}\ \left(\nabla w - \boldsymbol{\psi}\right)$$

$$\mathbf{Q} = \alpha\ \left(\nabla w - \boldsymbol{\psi}\right)$$

6.5 Verschiebungsgleichungen der Platte

Einsetzen der Elastizitätsgleichungen für die Biegung in das Momentengleichgewicht

$$\mathbf{L}^{\mathrm{T}}\,\mathbf{D}\,\mathbf{L}\,\boldsymbol{\psi} \;+\; \mathbf{Q} \;=\; \mathbf{0} \qquad (*)$$

6.5.1 KIRCHHOFF-Theorie

Bilde Gradienten:

$$\boldsymbol{\nabla}^{\mathrm{T}}\,\mathbf{L}^{\mathrm{T}}\,\mathbf{D}\,\mathbf{L}\,\boldsymbol{\psi} + \boldsymbol{\nabla}^{\mathrm{T}}\,\mathbf{Q} = \mathbf{0}$$

Einsetzen der Kräftegleichgewichtsbedingung $\boldsymbol{\nabla}^{\mathrm{T}}\,\mathbf{Q} + q = 0$ ergibt:

$$\boldsymbol{\nabla}^{T}\,\mathbf{L}^{T}\,\mathbf{D}\,\mathbf{L}\,\boldsymbol{\psi} = q$$

Zwangsbedingung für die schubstarre KIRCHHOFF-Platte $\boldsymbol{\psi} = \boldsymbol{\nabla} w$ liefert:

$$\boldsymbol{\nabla}^{\mathrm{T}}\,\mathbf{L}^{\mathrm{T}}\,\mathbf{D}\,\mathbf{L}\,\nabla w = q$$

Im Fall der isotropen elastischen Materialannahme folgt:

$$\begin{bmatrix} \dfrac{\partial^2}{\partial x_1^2} & \dfrac{\partial^2}{\partial x_2^2} & 2\dfrac{\partial^2}{\partial x_1\,\partial x_2} \end{bmatrix} \underbrace{\frac{E\,h^3}{12\left(1-\nu^2\right)}}_{=D} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \dfrac{\partial^2 w}{\partial x_1^2} \\ \dfrac{\partial^2 w}{\partial x_2^2} \\ 2\dfrac{\partial^2 w}{\partial x_1\,\partial x_2} \end{bmatrix} = q\left(x_1, x_2\right)$$

Als Ergebnis der Matrizenmultiplikation ergibt sich die biharmonische Plattengleichung der KIRCHHOFFschen Theorie:

$$\boxed{\frac{\partial^4 w}{\partial x_1^4} \;+\; \frac{\partial^4 w}{\partial x_2^4} \;+ 2\,\frac{\partial^4 w}{\partial x_1^2\;\partial x_2^2} \;=\; \frac{q}{D}}$$

Die DGL kann in eine Integralgleichung umgewandelt und diskretisiert werden. Die erforderlichen Verschiebungsansätze erfordern C^1 – Kontinuität, die schwierig zu gewährleisten ist.

Daher begnügt man sich auch für dünne Platten mit der REISSNER-MINDLIN-Theorie für schubweiche Platten, wozu nur C^0 – Stetigkeit erforderlich ist, wie sich zeigen wird.

6.5.2 REISSNER – MINDLIN – Theorie

Ersetze die Schubkräfte $\boldsymbol{Q}$ in der Momenten-Gleichung (*) durch die Elastizitätsbeziehung $\mathbf{Q} = \alpha\left(\nabla w - \boldsymbol{\psi}\right)$ für die isotrope, elastische Platte:

$$\mathbf{L}^{\mathrm{T}}\,\mathbf{D}\,\mathbf{L}\,\boldsymbol{\psi} \;+\; \boldsymbol{\alpha}\left(\nabla w - \boldsymbol{\psi}\right) = \mathbf{0} \qquad (**)$$

Ersetze Schubkräfte in der Kräftegleichgewichtsbedingung durch die Elastizitätsbeziehung für **Q** der isotropen, elastischen Platte:

$$\boldsymbol{\nabla}^{\mathrm{T}}\left[\alpha\left(\nabla w - \boldsymbol{\psi}\right)\right] \;+\; q \;=\; 0 \qquad (***)$$

(**) und (***) sind die irreduziblen Verschiebungsgleichungen für die schubweiche Platte in Form dreier partieller Differentialgleichungen 2. Ordnung. Sie müssen simultan gelöst werden, falls die Verschiebungen w nicht eliminiert werden, was zu Ableitungen 3. Ordnung führen würde. (siehe Vorlesung SPS).

Die Variationsgleichungen zu (**) und (***) können durch Konstruktion des gewichteten Residuums und anschließender partieller Integration oder aus der Variation der gesamten potentiellen Energie gewonnen werden.

Potentielle Energie:

$$\pi = \hat{\pi}(w, \boldsymbol{\psi}) = \pi_i - \pi_a = \frac{1}{2}\int_A (\mathrm{L}\boldsymbol{\psi})^{\mathrm{T}} \mathrm{D} (\mathrm{L}\boldsymbol{\psi}) dA + \frac{1}{2}\int_A (\nabla w - \boldsymbol{\psi})^{\mathrm{T}} \alpha (\nabla w - \boldsymbol{\psi}) \mathrm{d}A$$

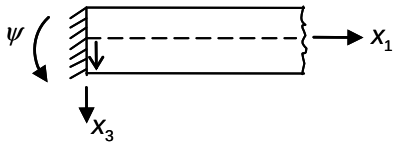
$$-\int_A w\, q \, \mathrm{d}A - \int_{\partial A_{\bar{Q}}} w\, Q_e \mathrm{d}s - \int_{\partial A_{\bar{M}}} \boldsymbol{\psi}^{\mathrm{T}}\, \mathbf{M}_e \mathrm{d}s \overset{!}{=} \text{Minimum}$$

$$\delta\pi = \frac{\partial \hat{\pi}}{\partial w}\delta\hat{w} + \frac{\partial \hat{\pi}}{\partial \boldsymbol{\psi}}\delta\boldsymbol{\psi} \overset{\mathbf{D}=\mathbf{D}^{\mathrm{T}}}{=} \int_A \delta\boldsymbol{\psi}^{\mathrm{T}} \mathbf{L}^{\mathrm{T}} \mathbf{D} \mathbf{L} \boldsymbol{\psi} \, \mathrm{d}A + \int_A (\delta\boldsymbol{\psi} - \nabla\delta w)^{\mathrm{T}} \alpha (\nabla w - \boldsymbol{\psi}) \mathrm{d}A -$$

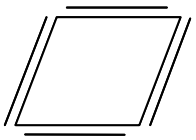
$$-\int_A \delta w\, q \, \mathrm{d}A - \int_{\partial A_{\bar{Q}}} \delta w \; Q_e \mathrm{d}s - \int_{\partial A_{\bar{M}}} \delta\boldsymbol{\psi}^{\mathrm{T}} \; \mathbf{M}_e \mathrm{d}s = 0$$

Randbedingungen

-eingespannt

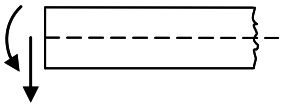


$w = 0$
$\psi = 0$

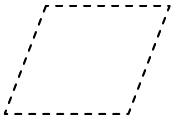


$w = 0,\ \psi_1 = 0,\ \psi_2 = 0$

- freier Rand:
  eingeprägte
  Kräfte und Momente

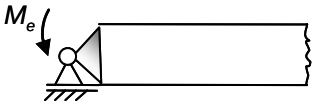


$Q = Q_e$



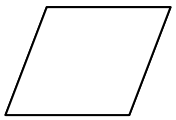
$M_n = M_{ne};\ M_s = M_{se};\ Q = Q_e$

-einfach gelenkiger Rand:



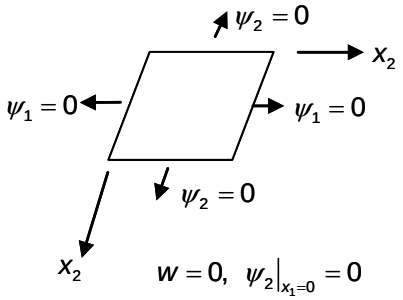
(„soft support“)

$w = 0$
$M = M_e$



$w = 0,\ M_n = M_{ne}$
$M_s = M_{se}$

(„hardsupport“)



$w = 0,\ \psi_2\big|_{x_1=0} = 0$

$\vdots$

Abb. 6.5-1: Randbedingungen der Platte

In der Praxis können Kombinationen aller 4 Randbedingungen auftreten.

$$\int_A \left[\delta\nabla w^{\mathrm{T}} \quad \delta\boldsymbol{\psi}^{\mathrm{T}}\right] \left\{ \begin{bmatrix} \boldsymbol{\alpha} & -\boldsymbol{\alpha} \\ -\boldsymbol{\alpha} & \boldsymbol{\alpha} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^{\mathrm{T}}\mathbf{D}\mathbf{L} \end{bmatrix} \right\} \begin{bmatrix} \nabla w \\ \boldsymbol{\psi} \end{bmatrix} \mathrm{d}A = \int_A \left[\delta w \quad \delta\boldsymbol{\psi}^{\mathrm{T}}\right] \begin{bmatrix} q \\ \mathbf{0} \end{bmatrix} \mathrm{d}A + \int_{\substack{\partial A_Q \\ \partial A_M}} \left[\delta w \quad \delta\boldsymbol{\psi}^{\mathrm{T}}\right] \begin{bmatrix} \mathrm{Q}_e \\ \mathbf{M}_e \end{bmatrix} \mathrm{d}s$$

In den Variationsgleichungen treten höchstens 1. Ableitungen der Verschiebung **w** auf.
Querschubsteifigkeit α und Biegesteifigkeit **D** werden addiert.

$$Gh + \frac{Eh^3}{12\left(1-\nu^2\right)}\frac{1}{l^2} = \frac{Eh}{12(1+\nu)}\left[5 + \frac{1}{1-\nu}\left(\frac{h}{l}\right)^2\right]$$

↑ aus $\mathbf{L}^{\mathrm{T}}\mathbf{D}\mathbf{L}$, wenn die Ableitung $\frac{\partial}{\partial x_1}$ und $\frac{\partial}{\partial x_1}$ entdimensioniert werden.

$x_1 = l\xi_1; \quad x_2 = l\xi_2$ und $\xi_1,\ \xi_2$ dimensionslos.

Mit kleiner werdender Plattendicke $h << l$ verschwindet der Term $\left(\frac{h}{l}\right)^2$ gegenüber 1

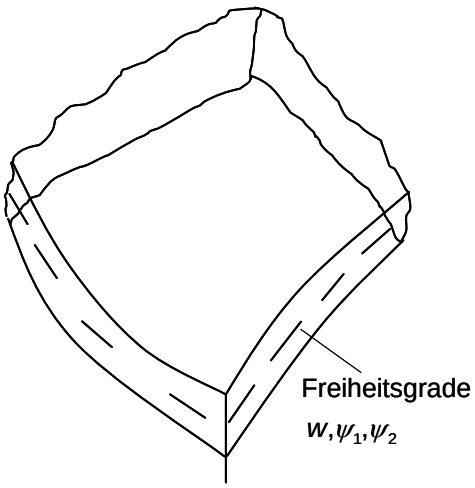
Bei der dünnen Platte $h << l$ wird die bezogene Plattensteifigkeit $\frac{D}{l^2}$ im Vergleich zur Schubsteifigkeit **Gh** klein.

→ Problem der schlechten Konditionierung von Gleichungssystemen.

6.6 Diskretisierung der Verschiebungs- und Rotationsfelder

Knotenvariable: $w^{\mathrm{k}},\ \psi_1^{\mathrm{k}},\ \psi_2^{\mathrm{k}}$

Diskretisierte Feldgrößen: $w(x_1,x_2) = \mathbf{N}_w(x_1,x_2)\,\mathbf{w}^{\mathrm{k}}$



$$\boldsymbol{\psi}(x_1,x_2) = \begin{bmatrix}\psi_1(x_1,x_2)\\ \psi_2(x_1,x_2)\end{bmatrix} = \begin{bmatrix}\mathbf{N}_{\psi_1}(x_1,x_2) & \mathbf{0}\\ \mathbf{0} & \mathbf{N}_{\psi_2}(x_1,x_2)\end{bmatrix}\begin{bmatrix}\boldsymbol{\psi}_1^{\mathrm{k}}\\ \boldsymbol{\psi}_2^{\mathrm{k}}\end{bmatrix}$$

Variation der diskretisierten Feldgrößen:

$$\delta w(x_1,x_2) = \mathbf{N}_w(x_1,x_2)\,\delta\mathbf{w}^{\mathrm{k}}$$

$$\delta\boldsymbol{\psi}(x_1,x_2) = \begin{bmatrix}\psi_1(x_1,x_2)\\ \psi_2(x_1,x_2)\end{bmatrix} = \begin{bmatrix}\mathbf{N}_{\psi_1}(x_1,x_2) & \mathbf{0}\\ \mathbf{0} & \mathbf{N}_{\psi_2}(x_1,x_2)\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\psi}_1^{\mathrm{k}}\\ \delta\boldsymbol{\psi}_2^{\mathrm{k}}\end{bmatrix}$$

Abb. 6.6-1: Freiheitsgrade des Plattenelements

Die Variationsgleichung lautet nach der Diskretisierung:

$$\int_A \begin{bmatrix} \nabla \mathbf{N}_w & \mathbf{0} \\ \mathbf{0} & \mathbf{N}_\psi \end{bmatrix}^{\mathrm{T}} \left\{ \begin{bmatrix} \boldsymbol{\alpha} & -\boldsymbol{\alpha} \\ -\boldsymbol{\alpha} & \boldsymbol{\alpha} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^{\mathrm{T}}\mathbf{D}\mathbf{L} \end{bmatrix} \right\} \begin{bmatrix} \nabla \mathbf{N}_w & \mathbf{0} \\ \mathbf{0} & \boldsymbol{N}_\psi \end{bmatrix} \mathrm{d}A \begin{bmatrix} \mathbf{w}^k \\ \boldsymbol{\psi}^k \end{bmatrix} =$$

$$= \int_A \begin{bmatrix} \mathbf{N}_w & \mathbf{0} \\ \mathbf{0} & \mathbf{N}_\psi \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} q \\ \mathbf{0} \end{bmatrix} \mathrm{d}A + \int_{\substack{\partial A_Q \\ \partial A_M}} \begin{bmatrix} \mathbf{N}_w & \mathbf{0} \\ \mathbf{0} & \mathbf{N}_\psi \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_e \\ \mathbf{M}_e \end{bmatrix} \mathrm{d}s$$

Einführen der Elementsteifigkeitsmatrizen:

$$\begin{bmatrix} \mathbf{k}_{ww} & \mathbf{k}_{w\psi} \\ \mathbf{k}_{w\psi} & \mathbf{k}_{\psi\psi} \end{bmatrix} \begin{bmatrix} \mathbf{w}^k \\ \boldsymbol{\psi}^k \end{bmatrix} = \begin{bmatrix} \mathbf{f}_w \\ \mathbf{f}_\psi \end{bmatrix}$$

Biegesteifigkeit: $\mathbf{k}_b = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{k}^b_{\psi\psi} \end{bmatrix}$ Querschubsteifigkeit: $\mathbf{k}_s = \begin{bmatrix} \mathbf{k}^s_{ww} & \mathbf{k}^s_{w\psi} \\ \mathbf{k}^s_{\psi w} & \mathbf{k}^s_{\psi\psi} \end{bmatrix}$

Untermatrizen :

$$\mathbf{k}^b_{\psi\psi} = \int_A \mathbf{N}^{\mathrm{T}}_\psi \mathbf{L}^{\mathrm{T}} \mathbf{D} \mathbf{L} \mathbf{N}_\psi \, \mathrm{d}A$$

$$\mathbf{k}^s_{ww} = \int_A \left(\nabla \mathbf{N}_w\right)^{\mathrm{T}} \boldsymbol{\alpha} \nabla \mathbf{N}_w \, \mathrm{d}A$$

$$\mathbf{k}^s_{w\psi} = \mathbf{k}^{s\,\mathrm{T}}_{\psi w} = -\int_A \left(\nabla \mathbf{N}_w\right)^{\mathrm{T}} \boldsymbol{\alpha} \mathbf{N}_\psi \, \mathrm{d}A$$

$$\mathbf{k}^s_{\psi\psi} = -\int_A \mathbf{N}^{\mathrm{T}}_\psi \, \boldsymbol{\alpha} \, \mathbf{N}_\psi \, \mathrm{d}A$$

$$\mathbf{f}_w = +\int_A \mathbf{N}^{\mathrm{T}}_w \, q \, \mathrm{d}A + \int_{\partial A_Q} \mathbf{N}^{\mathrm{T}}_w \, Q_e \mathrm{d}s$$

$$\mathbf{f}_\psi = \int_{\partial A_M} \mathbf{N}^{\mathrm{T}}_\psi \, \mathbf{M}_e \mathrm{d}s$$

Schrifttumsverzeichnis

1 „Elemente der Mathematik”, Herausg.: H. Griesel, H. Postel und F. Suhr, Schroedel Verlag 2002, S. 168.

2 Böhm, C. und G. Jacopini: „Flow Diagrams, Turing Machines and Languages with Formation Rules“, CACM (1966), 366-371.

3 G. Schmitt: “Fortran-90-Kurs, technisch orientiert. Einleitung in die Programmierung mit Fortran90“. R. Oldenbourg Verlag München, 1996.

4 H. Wehnes: „FORTRAN 77, Strukturierte Programmierung mit FORTRAN77“, Carl Hanser Verlag München, 1992, 7. überarbeitete Auflage.

5 „FORTRAN 90, Ein Nachschlagwerk“, RRZN Regionales Rechenzentrum für Niedersachsen / Universität Hannover, August 1993, 2. überarbeitete Auflage.