FINITE ELEMENT METHODE

Stationäre Wärmeströmung

Peter Jan Pahl

Berlin, im November 2000

Technische Universität Berlin
Institut für Bauingenieurwesen
Sekr. EB 3
Straße des 17. Juni 135
10623 Berlin

Prof. Dr.Dr.h.c.mult. Peter Jan Pahl

Tel. (030) 314 23193

Das Druckbild dieses Skriptes wurde von Frau Eva Ghosh gestaltet.

INHALTSVERZEICHNIS

1. AUFGABENSTELLUNG

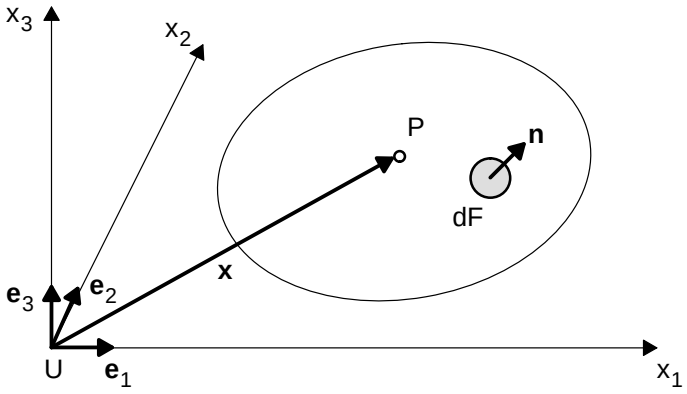
Gegeben sei ein ruhender Festkörper. Der Wärmefluß in diesem Körper sei unabhängig von der Zeit (stationär). An jedem Punkt des Körpers sei die Wärmemenge gegeben, die je Zeit- und Volumeneinheit zugeführt wird. An jedem Punkt der Oberfläche des Körpers sei entweder die Temperatur oder die Wärmemenge gegeben, die je Zeit- und Flächeneinheit durch die Oberfläche fließt. Zu bestimmen sind die Temperatur an denjenigen Punkten des Körpers, an denen sie nicht eingeprägt ist, und der Wärmefluß je Zeit- und Flächeneinheit an denjenigen Punkten der Oberfläche, an denen die Temperatur eingeprägt ist.

Die Lösung dieser Aufgabe wird in folgende Schnitte gegliedert :

– Mathematische Formulierung der physikalischen Aufgabe mit Differentialgleichungen
– Transformation der Differentialgleichungen in äquivalente Integralgleichungen mit einer für Näherungslösungen geeigneten Form (partielle Integration)
– Beschreibung der Geometrie des Festkörpers mit geometrischen finiten Elementen
– Beschreibung des Wärmeflusses im Festkörper mit physikalischen finiten Elementen
– Aufstellen algebraischer Systemgleichungen zur Bestimmung der Stützwerte des Lösungsansatzes
– Lösung der algebraischen Systemgleichungen (Profilmatix).

Für jeden dieser Schritte werden zunächst die theoretischen Grundlagen formuliert. Dann werden die Java-Klassen für Finite Elemente und für lineare Gleichungssysteme mit Profilstruktur definiert. Mit Objekten dieser Klassen wird ein Finite Element Modell für den Wärmefluß konstruiert. Die Algorithmen zur Berechnung des Wärmeflusses sind in diesem Modell implementiert. Das Skript enthält den vollständigen Code der Implementierung.

2. GEOMETRIE DES KÖRPERS



Der Festkörper wird in einem kartesischen globalen Koordinatensystem mit den Einsvektoren $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ in Richtung der Achsen x_1, x_2, x_3 beschrieben :

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Ein beliebiger Punkt P des Festkörpers wird relativ zum Ursprung U des Koordinatensystems durch seinen Ortsvektor **x** beschrieben :

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + x_3 \mathbf{e}_3 \tag{1}$$

x_i Koordinaten des Ortsvektors

Die Form des Festkörpers wird durch die Ortskoordinaten **x** der Punkte in seiner Oberfläche F festgelegt. Die Richtung der Oberfläche am Punkt **x** wird durch die Außennormale **n** des Körpers beschrieben :

$$\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = n_1 \mathbf{e}_1 + n_2 \mathbf{e}_2 + n_3 \mathbf{e}_3 \tag{2}$$

n_i Koordinaten der Außennormale

Im folgenden wird vorausgesetzt, daß die Außennormale **n** ein Einheitsvektor ist. Dann ist der inkrementelle Flächenvektor d**f** der Oberfläche am Punkt **x** das Produkt des Flächenbetrages dF und der Normalen **n** :

$$d\mathbf{f} = \mathbf{n}\, dF \quad \wedge \quad \mathbf{n}^T\mathbf{n} = 1 \tag{3}$$

dF Betrag des inkrementellen Flächenvektors d**f**

Das Volumen V des Körpers ist die Menge der Punkte im Inneren der Oberfläche F.

3. PHYSIK DES KÖRPERS

Zustandsvariable : Der Zustand des Körpers wird durch die Temperatur des Stoffes an jedem Punkt **x** des Körpers beschrieben. Da ein stationärer Zustand vorausgesetzt wird, ist die Wärme je Volumeneinheit konstant. Sie geht nicht als Variable in die Formulierung der Aufgabe ein. Durch die Oberfläche des Körpers fließt Wärme. Wegen des stationären Zustandes ist die Wärme konstant, die je Zeit- und Flächeneinheit durch die Oberfläche fließt. Ein Wärmefluß von innen nach außen (also in Richtung der Oberflächennormale **n**) wird als positiv definiert und als Wärmeabfluß bezeichnet.

$u(\mathbf{x})$ Temperatur am Punkt $\mathbf{x} \in V$

$q(\mathbf{x})$ Wärmeabfluß je Zeit- und Flächeneinheit am Punkt $\mathbf{x} \in F$

Einwirkungen : An jedem Punkt **x** im Volumen des Körpers ist der Wärmezufluß $w_0(\mathbf{x})$ je Zeit- und Volumeneinheit gegeben. In einem Teil $F_1 \subseteq F$ der Oberfläche ist an jedem Punkt **x** die Temperatur $u_0(\mathbf{x})$ eingeprägt. In dem verbleibenden Teil $F_2 \subseteq F$ der Oberfläche ist an jedem Punkt **x** der Wärmeabfluß q_0 je Zeit- und Flächeneinheit eingeprägt.

$w_0(\mathbf{x})$ Wärmezufluß je Zeit- und Volumeneinheit am Punkt $\mathbf{x} \in V$

$u_0(\mathbf{x})$ eingeprägte Temperatur am Punkt $\mathbf{x} \in F_1$

$q_0(\mathbf{x})$ eingeprägter Wärmeabfluß je Zeit- und Flächeneinheit am Punkt $\mathbf{x} \in F_2$

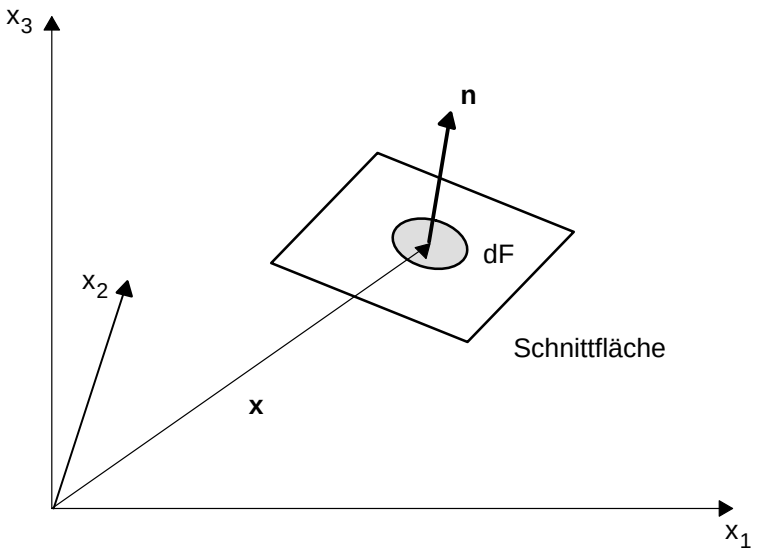
Da an jedem Punkt der Oberfläche entweder die Temperatur oder der Wärmeabfluß je Zeit- und Flächeneinheit eingeprägt ist, gilt $F_1 \cup F_2 = F$. Der Wärmeabfluß darf nur dann an allen Punkten der Oberfläche F eingeprägt sein, wenn die gleiche Wärmemenge im Inneren des Körpers zugeführt und durch die Oberfläche abgeführt wird, da sonst der Zustand nicht stationär ist. Im allgemeinen wird auf einem Teil der Oberfläche die Temperatur eingeprägt.

Temperaturgradient : Der Gradient des Temperaturfeldes u(**x**) an einem Punkt **x** des Körpers heißt der Temperaturgradient am Punkt **x** und wird mit **g** bezeichnet. Der Temperaturgradient wird als Produkt des Operatorvektors **d** mit dem Skalar u dargestellt.

$$\mathbf{g} = \mathbf{d}u \tag{4}$$

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x_1} \\ \frac{\partial u}{\partial x_2} \\ \frac{\partial u}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{bmatrix} u$$

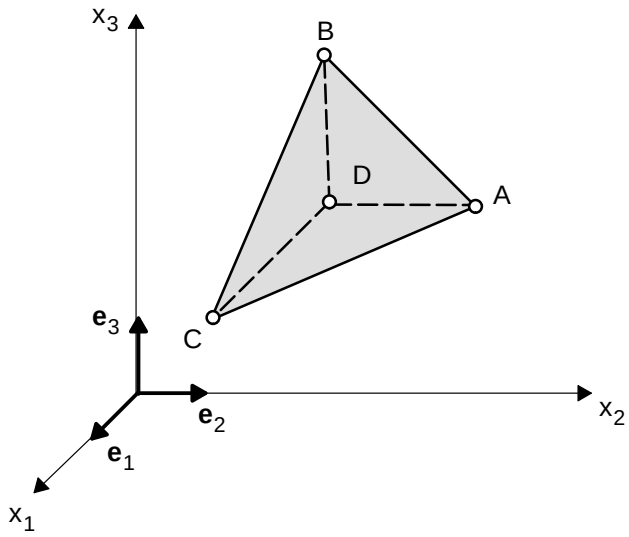
Wärmestrom : Gegeben sei eine ebene Schnittfläche mit der Normalen **n** durch einen Punkt **x** des Körpers. Die Wärmemenge, die je Zeiteinheit in der positiven Richtung von **n** durch ein Flächenelement mit Betrag dF am Punkt **x** fließt, sei dW. Dann heißt der Grenzwert des Verhältnisses dW/dF der Wärmestrom am Punkt **x** durch die Fläche mit der Normalen **n** und wird mit q bezeichnet.



$$q = \lim_{dF \to 0} \frac{dW}{dF} \tag{5}$$

q Betrag des Wärmestroms

Wärmezustand : Der Wärmestrom an einem Punkt **x** ist von der Normalen **n** der betrachteten Schnittfläche abhängig. Es stellt sich die Frage, ob die Wärmeströme durch verschiedene Schnittflächen am Punkt **x** voneinander abhängig sind. Zur Beantwortung dieser Frage wird ein infinitesimales Tetraeder ABCD am Punkt **x** betrachtet. Drei der Seitenflächen des Tetraeders sind normal zu den Einheitsvektoren $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. Die vierte Seitenfläche ABC besitzt den Betrag dF und die Einheitsnormale **n**.



Die Teiloberflächen des Tetraeders besitzen folgende Flächenvektoren :

Seite	Normale	Betrag	Flächenvektor
ABC	$\mathbf{n}$	dF	$\mathbf{n}\,dF = (n_1\mathbf{e}_1 + n_2\mathbf{e}_2 + n_3\mathbf{e}_3)\,dF$
ADB	$-\mathbf{e}_1$	dF_1	$-\mathbf{e}_1 dF_1$
BDC	$-\mathbf{e}_2$	dF_2	$-\mathbf{e}_2 dF_2$
CDA	$-\mathbf{e}_3$	dF_3	$-\mathbf{e}_3 dF_3$

Da die Oberfläche des Tetraeders geschlossen ist, ist die Summe der vier Flächenvektoren gleich null :

$$\mathbf{n}\,dF - \mathbf{e}_1 dF_1 - \mathbf{e}_2 dF_2 - \mathbf{e}_3 dF_3 = \mathbf{0}$$

$$(n_1\mathbf{e}_1 + n_2\mathbf{e}_2 + n_3\mathbf{e}_3)\,dF = \mathbf{e}_1 dF_1 + \mathbf{e}_2 dF_2 + \mathbf{e}_3 dF_3$$

$$dF_i = n_i\,dF \quad \text{für} \quad i = 1,2,3 \tag{6}$$

Der Wärmestrom durch die Tetraederfläche normal zum Achsvektor $\mathbf{e}_i$ sei f_i in der positiven Richtung von $\mathbf{e}_i$. Der Wärmestrom durch die Fläche ABC sei q in Richtung der Normalen **n**. Da im Tetraeder keine Wärme gespeichert wird, ist die Summe der Wärmezuflüsse gleich null :

$$q\,dF - f_1 dF_1 - f_2 dF_2 - f_3 dF_3 = 0$$

$$q = n_1 f_1 + n_2 f_2 + n_3 f_3$$

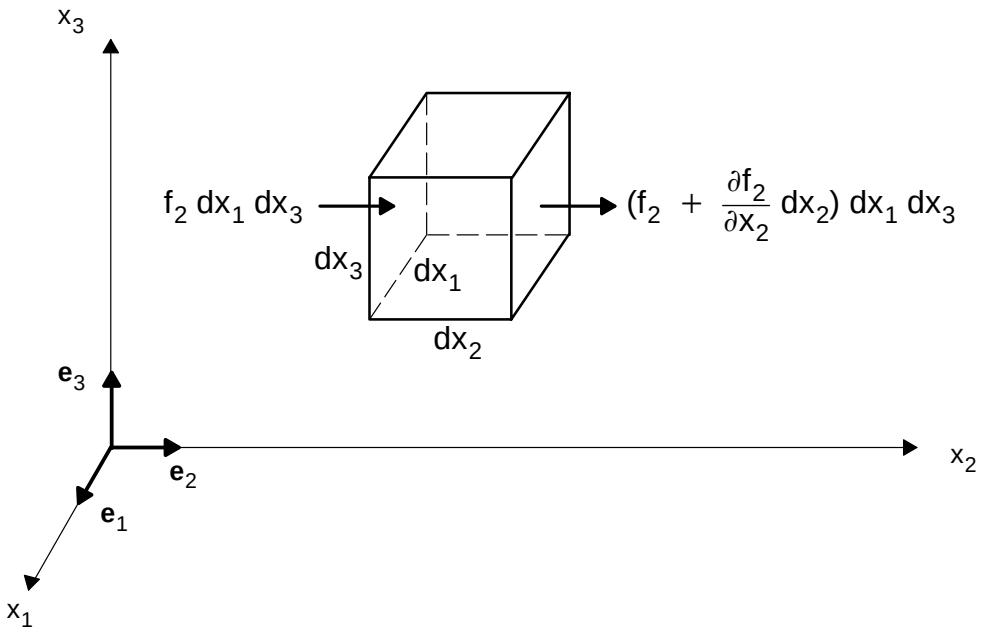
$$q = \mathbf{n}^T \mathbf{f} \qquad (7)$$

Der Vektor **f** mit den Koordinaten f_i heißt der Wärmezustand des Körpers am Punkt **x**. Ist der Wärmezustand bekannt, so kann der Wärmestrom q durch eine beliebige Schnittfläche mit der Normalen **n** am Punkt **x** mit der Formel (7) berechnet werden.

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = f_1 \mathbf{e}_1 + f_2 \mathbf{e}_2 + f_3 \mathbf{e}_3 \qquad (8)$$

f	Wärmezustand am Punkt **x**
f_i	Wärmefluß durch die Schnittfläche normal zu $\mathbf{e}_i$ in der Richtung der Flächennormale $\mathbf{e}_i$

Wärmebilanz : Der Wärmezustand **f** des Körpers ist eine Funktion des Ortsvektors **x**. Zur Bestimmung der Änderung von **f** mit **x** wird ein infinitesimales Volumenelement dV des Körpers mit Kantenlängen dx_i betrachtet, dessen Seitenflächen normal zu den Achsvektoren $\mathbf{e}_i$ sind.



In dem Volumenelement dV wird keine Wärme gespeichert. Die Summe des Wärmezuflusses im Inneren und der Wärmezuflüsse durch die Seiten ist also null. Der Wärmefluß durch die Seiten ist exemplarisch für die Richtung x_2 gezeigt. Der Betrag der Seitenfläche ist $dx_1\,dx_3$. Durch die linken Seitenflächen fließt Wärme zu, durch die rechte Seite fließt Wärme ab. Die Wärmebilanz für den Wärmezufluß durch die beiden Seiten ist daher :

$$f_2\,dx_1 dx_3 - \left(f_2 + \frac{\partial f_2}{\partial x_2}\,dx_2\right) dx_1\,dx_3 = -\frac{\partial f_2}{\partial x_2}\,dx_1\,dx_2 dx_3$$

Betrachtet man alle Seiten des Volumenelementes sowie den Wärmezufluß im Inneren, so gilt folgende Wärmebilanz :

$$-\frac{\partial f_1}{\partial x_1}\,dx_1\,dx_2\,dx_3 - \frac{\partial f_2}{\partial x_2}\,dx_1\,dx_2\,dx_3 - \frac{\partial f_3}{\partial x_3}\,dx_1\,dx_2\,dx_3 + w_0\,dx_1\,dx_2\,dx_3 = 0$$

$$\frac{\partial f_1}{\partial x_1} + \frac{\partial f_2}{\partial x_2} + \frac{\partial f_3}{\partial x_3} = w_0 \qquad (9)$$

Mit dem in Formel (4) definierten Operatorvektor **d** kann die Wärmebilanzgleichung (9) vektoriell geschrieben werden :

$$\mathbf{d}^T\mathbf{f} = w_0$$

(10)

d Operatorvektor, siehe (4)

f Wärmezustand, siehe (8)

Stoffgesetz : Experimentell wird beobachtet, daß Wärme von Regionen höherer Temperatur zu Regionen niedrigerer Temperatur fließt. Es gibt also einen Zusammenhang zwischen dem Wärmezustand **f** und dem Temperaturgradienten **g**. Dieser Zusammenhang wird durch Messung bestimmt. Für lineares Stoffverhalten gilt folgende Beziehung :

$$\mathbf{f} = -\mathbf{C}\,\mathbf{g} \qquad (11)$$

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = - \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} * \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}$$

C Wärmeleitmatrix

c_{im} Wärmeleitkoeffizienten

Das Minuszeichen wird eingeführt, damit der Wärmeleitkoeffizient für eindimensionale Wärmeströmung positiv ist. Für isotropes Materialverhalten genügt die Messung des eindimensionalen Wärmeleitkoeffizienten c :

$$\mathbf{C}_{iso} = \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{bmatrix} \tag{12}$$

Randbedingungen : Die Menge der Punkte der Oberfläche F, an denen die Temperatur u_0 eingeprägt ist, sei F_1. Die Menge der Punkte, an denen der Wärmeabfluß q_0 eingeprägt ist, sei F_2. Dann müssen folgende Randbedingungen befriedigt sein :

$\mathbf{x} \in F_1$: $u = u_0$ (13)

$u_0(\mathbf{x})$ eingeprägte Temperatur am Punkt **x**

$\mathbf{x} \in F_2$: $q = q_0$ (14)

$q_0(\mathbf{x})$ eingeprägter Wärmeabfluß am Punkt **x**

q Wärmestrom in Richtung der Außennormalen **n**

Durch Substitution der Beziehung (7) zwischen dem Wärmestrom q und dem Wärmezustand **f** folgt aus (14) :

$\mathbf{x} \in F_2$: $\mathbf{n}^T\mathbf{f} = q_0$ (15)

Bestimmungsgleichungen : Die im vorangehenden aufgestellten Bestimmungsgleichungen für den Wärmefluß werden im folgenden zusammengestellt :

$$\mathbf{g} = \mathbf{d}u \qquad \mathbf{x} \in V \qquad (4)$$

$$q = \mathbf{n}^T \mathbf{f} \qquad \mathbf{x} \in F \qquad (5)$$

$$\mathbf{d}^T \mathbf{f} = w_0 \qquad \mathbf{x} \in V \qquad (10)$$

$$\mathbf{f} = -\mathbf{C}\mathbf{g} \qquad \mathbf{x} \in V \qquad (11)$$

u Temperatur

q Wärmestrom durch Fläche mit Normale $\mathbf{n}$

$\mathbf{g}$ Temperaturgradient

$\mathbf{n}$ Flächennormale mit $\mathbf{n}^T \mathbf{n} = 1$

$\mathbf{f}$ Wärmezustand

w_0 eingeprägte Wärmezufluß je Zeit- und Volumeneinheit

$\mathbf{C}$ Wärmeleitmatrix

$\mathbf{d}$ Operatorvektor

Auf der Oberfläche des Körpers gelten folgende Randbedingungen :

$$u = u_0 \qquad \mathbf{x} \in F_1 \qquad (13)$$

$$\mathbf{n}^T \mathbf{f} = q_0 \qquad \mathbf{x} \in F_2 \qquad (14)$$

u_0 eingeprägte Temperatur

q_0 eingeprägter Wärmeabfluß

4. INTEGRALFORM DER BESTIMMUNGSGLEICHUNGEN

Methode der gewichteten Reste : Für die spätere numerische Approximation ist es vorteilhaft, die Differentialgleichungen der stationären Wärmeflußaufgabe durch Integralgleichungen zu ersetzen. Dazu verwendet man die Methode der gewichteten Reste. Diese Methode gliedert sich in die Schritte, die in den folgenden Absätzen beschrieben sind.

Lösungsansatz : Für den Temperaturverlauf im Körper V wird ein Ansatz u(**x**) und für den Wärmestromverlauf auf der Oberfläche F ein Ansatz q(**x**) gewählt. Ein Ansatz besteht aus linear unabhängigen Funktionen des Ortsvektors **x**, die mit zunächst unbekannten Skalaren multipliziert und addiert werden. Die Skalaren heißen freie Parameter des Ansatzes.

$$u(\mathbf{x}) = \sum_i u_i\, s_i(\mathbf{x}) \qquad \mathbf{x} \in V \tag{15}$$

$$q(\mathbf{x}) = \sum_m q_m t_m(\mathbf{x}) \qquad \mathbf{x} \in F \tag{16}$$

$s_i(\mathbf{x})$ i-te Ansatzfunktion für die Temperatur u

$t_m(\mathbf{x})$ m-te Ansatzfunktion für den Wärmefluß q

u_i freie Parameter des Temperaturansatzes

q_m freie Parameter des Ansatzes für den Wärmeabfluß

Reste : Die Ansätze (15) und (16) werden in die Bestimmungsgleichungen (4), (5), (10), (11), (13) und (14) für den stationären Wärmefluß substituiert. Da die frei gewählten Ansätze im allgemeinen keine exakte Lösung der Aufgabe sind, befriedigen sie die Bestimmungsgleichungen nicht exakt. Die resultierenden Reste werden wie folgt definiert :

$$\mathbf{r}_1 = \mathbf{g} - \mathbf{d}\,u \qquad \mathbf{x} \in V \tag{17}$$

$$\mathbf{r}_2 = \mathbf{f} + \mathbf{C}\,\mathbf{g} \qquad \mathbf{x} \in V \tag{18}$$

$$r_3 = q - \mathbf{n}^T\mathbf{f} \qquad \mathbf{x} \in F \tag{19}$$

$$r_4 = \mathbf{d}^T\mathbf{f} - w_0 \qquad \mathbf{x} \in V \tag{20}$$

$$r_5 = u_0 - u \qquad \mathbf{x} \in F_1 \tag{21}$$

$$r_6 = q_0 - q \qquad \mathbf{x} \in F_2 \tag{22}$$

A priori Bedingungen : Ein Teil der Bestimmungsgleichungen wird mit dem Ansatz exakt befriedigt. Diese Gleichungen heißen die a priori (erfüllten) Bedingungen. Die entsprechenden Reste in (17) bis (22) sind null. Die Wahl der a priori Bedingungen bestimmt die Integralform der Bestimmungsgleichungen für den Wärmefluß. Im folgenden wird die Integralform für folgende a priori Bedingungen aufgestellt :

$$\mathbf{g} = \mathbf{d}u = \sum_i u_i \, \mathbf{d}\,(s_i(\mathbf{x}) \quad \Rightarrow \quad \mathbf{r}_1 = 0 \qquad (17')$$

$$\mathbf{f} = -\mathbf{C}\,\mathbf{g} \quad \Rightarrow \quad \mathbf{r}_2 = 0 \qquad (18')$$

Gewichtete Reste : Die Reste r_3 bis r_6 der Bestimmungsgleichungen, die nicht a priori erfüllt sind, werden mit einer Gewichtsfunktion multipliziert und über ihren Gültigkeitsbereich integriert. Die Gewichtsfunktionen werden so gewählt, daß die physikalischen Einheiten der Terme der resultierenden Integralgleichung gleich sind :

– Der Rest r_3 der Wärmestromgleichung wird mit beliebigen Variationen $\delta u(\mathbf{x})$ der Temperatur multipliziert und über die Fläche F des Körpers integriert.

– Der Rest r_4 der Wärmebilanzgleichung wird mit beliebigen Variationen $\delta u(\mathbf{x})$ der Temperatur multipliziert und über das Volumen V des Körpers integriert.

– Der Rest r_5 der Temperaturrandbedingung wird mit beliebigen Variationen $\delta q(\mathbf{x})$ des Wärmestroms q (von innen nach außen) multipliziert und über die Teiloberfläche F_1 des Körpers integriert.

– Der Rest r_6 der Wärmestromrandbedingung wird mit beliebigen Variationen $\delta u(\mathbf{x})$ der Temperatur multipliziert und über die Teilfläche F_2 des Körpers integriert.

$$\int_V \delta u \; (\mathbf{d}^T\mathbf{f} - w_0)\, dV + \int_{F_1} \delta q \,(u_0 - u)\, dF + \int_F \delta u \; (q - \mathbf{n}^T\mathbf{f})\, dF + \int_{F_2} \delta u \; (q_0 - q)\, dF = 0 \qquad (23)$$

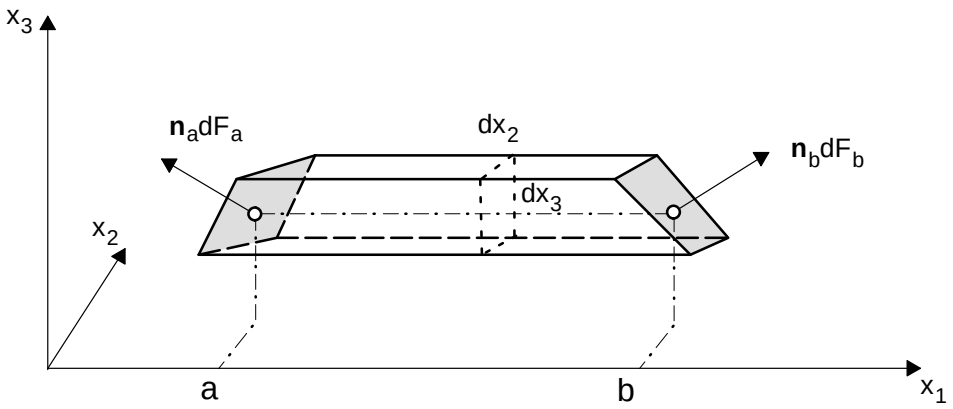
Exakte Lösung : Die Ansätze (15) und (16) sind die exakte Lösung der Aufgabe, wenn die Gleichung (23) für beliebige Variationen $\delta u(\mathbf{x})$ und $\delta q(\mathbf{x})$ befriedigt ist. Befriedigen die Ansätze nämlich eine Bestimmungsgleichung an mindestens einem Punkt **a** nicht, so wird $\delta u(\mathbf{a})$ oder $\delta q(\mathbf{a})$ gleich 1 gewählt und $\delta u = \delta q = 0$ für $\mathbf{x} \neq \mathbf{a}$: die Gleichung (23) ist nicht befriedigt. Befriedigen die Ansätze alle Bestimmungsgleichungen, so enthält jeder Term in (23) unabhängig von der Wahl von δu und δq einen Faktor null : die Gleichung (23) ist befriedigt.

Satz von Gauss : Aus der Regel für die partielle Ableitung eines Produktes $v(\mathbf{x})w(\mathbf{x})$ nach der Koordinate x_1 folgt durch Integration über das Volumen V des Körpers :

$$\frac{\partial}{\partial x_1}(vw) = \frac{\partial v}{\partial x_1} w + v\frac{\partial w}{\partial x_1}$$

$$\int_V v \frac{\partial w}{\partial x_1} dV = -\int_V \frac{\partial v}{\partial x_1} w\, dV + \int_V \frac{\partial (vw)}{\partial x_1} dx_1\, dx_2\, dx_3 \qquad (24)$$

Der Körper V wird in Stäbe mit Achse parallel zur x_1-Achse und infinitesimalem Querschnitt $dx_2 dx_3$ zerlegt. Die Achse eines typischen Stabes schneidet die Oberfläche F des Körpers bei $x = a$ und $x = b$. Der Stab schneidet die Flächenelemente $n_a dF_a$ und $n_b dF_b$ aus der Oberfläche F. Dann gilt :



$$\int_a^b \left\{ \frac{\partial (vw)}{\partial x_1} dx_2\, dx_3 \right\} dx_1 = vw\, dx_2\, dx_3 \Big|_a^b$$

Die Werte von $dx_2 dx_3$ bei x = b und x = a werden substituiert :

$$x = b : \quad dx_2\, dx_3 = \mathbf{e}_1 \cdot (\mathbf{n}_b\, dF_b) = n_{b1}\, dF_b$$

$$x = a : \quad dx_2\, dx_3 = \mathbf{e}_1 \cdot (-\mathbf{n}_a\, dF_a) = -n_{a1}\, dF_a$$

$$\int_a^b \left\{ \frac{\partial (vw)}{\partial x_1} dx_2\, dx_3 \right\} dx_1 = (vw\, n_1 dF)_b + (vw n_1 dF)_a$$

Summiert man die Beiträge aller Stäbe, so folgt für den Körper V mit (24) :

$$\int_V \frac{\partial (vw)}{\partial x_1} dV = \sum \left\{ (vw n_1 dF)_b + (vw n_1 dF)_a \right\} = \int_F vw\, n_1 dF$$

$$\int_V v \frac{\partial w}{\partial x_1} dV = -\int_V \frac{\partial v}{\partial x_1} w dV + \int_F vw\, n_1 dF \qquad (25)$$

Ähnliche Formeln gelten für die partiellen Ableitungen nach x_2 und x_3.

Partielle Integration : Der Satz von Gauss wird zur partiellen Integration des ersten Terms der Gleichung (23) benutzt :

$$\int_V \delta u\ \mathbf{d}^T\mathbf{f}\ dV = \int_V \delta u\ \left(\frac{\partial f_1}{\partial x_1} + \frac{\partial f_2}{\partial x_2} + \frac{\partial f_3}{\partial x_3}\right) dV$$

$$= -\int_V \left\{\frac{\partial(\delta u)}{\partial x_1} f_1 + \frac{\partial(\delta u)}{\partial x_2} f_2 + \frac{\partial(\delta u)}{\partial x_3} f_3\right\} dV$$

$$+ \int_F \delta u\ (f_1 n_1 + f_2 n_2 + f_3 n_3)\ dF$$

$$\int_V \delta u\ \mathbf{d}^T\mathbf{f}\ dV = -\int_V \delta\mathbf{g}^T\ \mathbf{f}\ dV + \int_F \delta u\ \mathbf{n}^T\mathbf{f}\ dF \qquad (26)$$

Das Ergebnis (26) wird in (23) substituiert :

$$-\int_V \delta\mathbf{g}^T\ \mathbf{f}\ dV + \int_F \delta u\ \mathbf{n}^T\mathbf{f}\ dF - \int_V \delta u\ w_o\ dV +$$

$$\int_{F_1} \delta q\ (u_0 - n)\ dF + \int_F \delta u\ (q - n^T f)\ dF + \int_{F_2} \delta u\ (q_0 - q)\ dF = 0$$

$$-\int_V \delta\mathbf{g}^T\mathbf{f}\ dV - \int_V \delta u\ w_0\ dV + \int_{F_1} \delta q\ (u - u_0)\ dF$$

$$+ \int_{F_1} \delta u\ q\ dF + \int_{F_2} \delta u\ q_0\ dF = 0$$

Integralform des stationären Wärmeflusses :

Auf F_1 wird $u = u_0$ gesetzt. Dann führt die Substitution von $\mathbf{f} = -\mathbf{Cg}$ in die vorangehende Gleichung zu der gesuchten Integralform :

$$\int_V \delta\mathbf{g}^T\ \mathbf{Cg}\ dV = \int_V \delta u\ w_0\ dV - \int_{F_1} \delta u\ q\ dF - \int_{F_2} \delta u\ q_0\ dF \qquad (27)$$

$$\mathbf{g} = \mathbf{d}\,u \qquad \mathbf{x} \in V \qquad (17)$$

$$\mathbf{f} = -\mathbf{C}\,\mathbf{g} \qquad \mathbf{x} \in V \qquad (18)$$

$$q = \mathbf{n}^T\mathbf{f} \qquad \mathbf{x} \in F \qquad (19)$$

$$u = u_0 \qquad \mathbf{x} \in F_1 \qquad (28)$$

Die a priori erfüllten Bedingungen (17) bis (19) sind Bestandteil der Formulierung des stationären Wärmeflusses als Integralaufgabe.

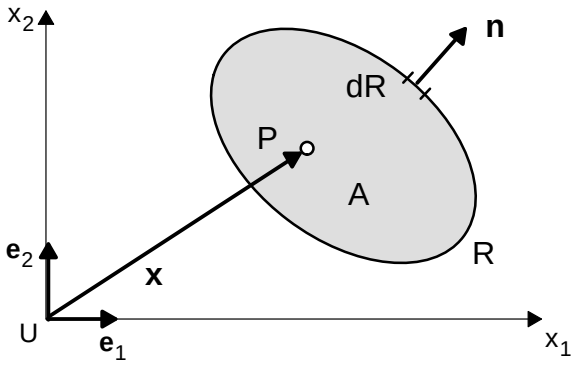
5. FINITE APPROXIMATION DER WÄRMESTRÖMUNG

5.1 EINFÜHRUNG

Geometrische Approximation : Die Wärmeströmung ist häufig in Gebieten mit unregelmäßiger Form zu bestimmen. Die Oberfläche solcher Gebiete läßt sich im allgemeinen nicht zweckmäßig mit einer einzigen Gleichung beschreiben, wie dies im Sonderfall der Kugel mit $x_1^2 + x_2^2 + x_3^2 = a^2$ noch möglich ist. Daher wird die Oberfläche durch eine aus einfachen Teilflächen zusammengestzte Fläche approximiert, beispielsweise durch ein Polyeder mit dreieckigen Facetten. Jede der Facetten heißt ein finites Flächenelement, da seine Abmessungen endlich sind.

Physikalische Approximation : Zur Lösung der Bestimmungsgleichungen für den Wärmefluß benötigt man Lösungsansätze. Für Gebiete mit unregelmäßiger Form stehen im allgemeinen keine Funktionen des Ortsvektors **x** zur Verfügung, die zur Approximation des Verlaufs der Verhaltensvariablen an allen Punkten des Lösungsgebietes geeignet sind. Daher wird das Lösungsgebiet in finite Elemente (Elemente mit endlichen Abmessungen) mit einfacher Form unterteilt, beispielsweise in Tetraeder. Für jedes der finiten Elemente gibt es einfache Ansatzfunktionen, deren Wert außerhalb des Elementes auf null gesetzt wird. Der Gesamtansatz für das Lösungsgebiet ist dann die Summe der Elementansätze über alle Elemente.

Ebene Scheibe : Die Entwicklung der Finite Element Methode für dreidimensionale Körper erfordert räumliche Darstellungen. Diese werden im folgenden vermieden, indem die Wärmeströmung in einer dünnen Scheibe mit der Höhe 1.0 betrachtet wird. Die Mittelfläche $x_3 = 0.0$ der Scheibe wird mit A bezeichnet, der Rand der Mittelfläche mit R.



R_1 Teil des Randes, auf dem die Temperatur eingeprägt ist

R_2 Teil des Randes, auf dem der Wärmefluß eingeprägt ist

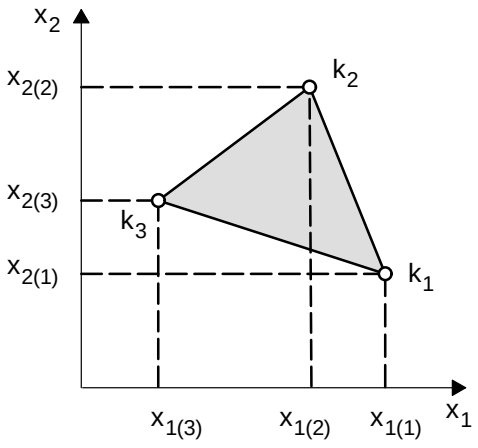
Die Oberflächen $x_3 = -0.5$ und $x_3 = 0.5$ der Scheibe seien perfekt isoliert. Das Material der Scheibe sei homogen und die Einwirkungen w_0, u_0, q_0 seien von x_3 unabhängig. Dann sind die Temperatur u und der Wärmestrom q unabhängig von x_3. Folglich kann der Wärmefluß in der Scheibe als zweidimensionale Aufgabe behandelt werden. Die Integralgleichung (27) erhält für den Wärmefluß in der Scheibe folgende Form :

$$\int_A \delta \mathbf{g}^T \mathbf{C}\, \mathbf{g}\, dA = \int_A \delta u w_0 dA - \int_{R_1} \delta u q dR - \int_{R_2} \delta u\, q_0 dR \qquad (29)$$

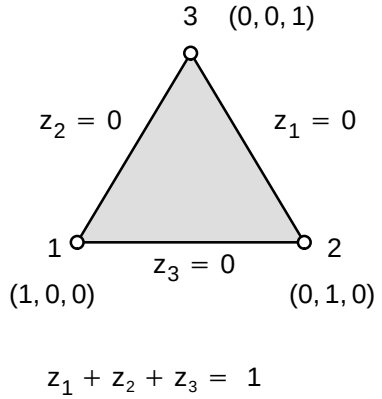
Die in der Formulierung enthaltenen Vektoren und Matrizen sind jetzt zweidimensional und nur von den Koordinaten x_1, x_2 abhängig.

5.2 FINITE GEOMETRIE

Koordinatensysteme : Das finite Scheibenelement wird in dem globalen Koordinatensystem (x_1, x_2) und in dem normalisierten Koordinatensystem (z_1, z_2, z_3) dargestellt. Im globalen System sind die Koordinaten $(x_{1(i)}, x_{2(i)})$ der Eckpunkte i = 1,2,3 für ein bestimmtes Dreieck gegeben. Im normalisierten System sind die Koordinaten (1,0,0), (0,1,0) und (0,0,1) der Eckpunkte fest vereinbart und für alle Dreiecke gleich. Finite Elemente können auch andere Formen besitzen, beispielsweise die eines Rechtecks mit 4 Knoten oder die eines krummlinigen Dreiecks mit 6 Knoten.



globales Koordinatensystem



normalisiertes Koordinatensystem

Topologische Abbildung : Die Knoten des Dreiecks werden mit natürlichen Zahlen k_1, k_2, k_3 identifiziert. Die Reihenfolge der Knoten definiert die Richtung des Flächenvektors und damit das Vorzeichen des Flächenbetrages. Im folgenden wird vorausgesetzt, daß die Knotenfolge k_1, k_2, k_3 auf dem Rand gegen den Uhrzeigersinn läuft und zu einem positiven Betrag der Fläche führt. Die Zahlen k_1, k_2, k_3 heißen globale Knotennummern.

Im normalisierten Koordinatensystem besitzen die Knoten immer die lokalen Knotennummern 1,2,3, deren Reihenfolge auf dem Rand gegen den Uhrzeigersinn läuft. Der Zusammenhang zwischen den Knoten im globalen und im normalisierten Koordinatensystem wird durch eine topologische Abbildung t : K → N festgelegt :

$$t : \{k_1, k_2, k_3\} \rightarrow \{1, 2, 3\} \qquad \text{mit} \quad t(k_i) = i$$

Geometrische Interpolation : Die Form eines finiten Elementes ist durch die Koordinaten seiner Knoten und die Interpolationspolynome für seine Koordinaten festgelegt. Die Interpolationspolynome sind Funktionen der normalisierten Koordinaten z_1, z_2, z_3. Für das gezeigte Dreieck sind die Interpolationspolynome linear, im allgemeinen jedoch nicht. Für die Interpolation einer Koordinate x_i gilt :

$$x_i = \mathbf{x}_i^T \mathbf{s} \qquad i = 1,2$$

(30)

$$\begin{bmatrix} x_i \end{bmatrix} = \begin{bmatrix} x_{i(1)} & x_{i(2)} & x_{i(3)} \end{bmatrix} * \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

$\mathbf{x}_i$ Stützvektor für Koordinate x_i

$\mathbf{s}(\mathbf{z})$ Formvektor

Im vorliegenden Dreieckelement sind die Formvektoren für x_1 und x_2 gleich und werden daher beide mit **s** bezeichnet. Der Ortsvektor **x** wird wie folgt interpoliert :

$$\mathbf{x} = \mathbf{X}\,\mathbf{s} \qquad (31)$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_{1(1)} & x_{1(2)} & x_{1(3)} \\ x_{2(1)} & x_{2(2)} & x_{2(3)} \end{bmatrix} * \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

X Koordinatenmatrix

$\mathbf{s}(\mathbf{z})$ Formvektor

Normalisierte Ableitungen des Formvektors : Die Ableitung des Formvektors **s** nach der Koordinate z_i heißt i-te normalisierte Ableitung des Formvektors und wird mit $\mathbf{s}_i$ bezeichnet. Für das lineare Dreieck sind die Ableitungen $\mathbf{s}_i$ Einheitsvektoren. Im allgemeinen sind die Ableitungen Funktionen $\mathbf{s}_i(\mathbf{z})$ der normalisierten Koordinaten. Sie werden spaltenweise in einer Matrix $\mathbf{S}_z$ angeordnet :

$$\mathbf{s}_i = \frac{\partial \mathbf{s}}{\partial z_i} \qquad \mathbf{s}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{s}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{s}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{32}$$

$$\mathbf{S}_z = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Normalisierte Ableitungen der globalen Koordinaten : Die partielle Ableitung der globalen Koordinaten x_i nach der normalisierten Koordinate z_k wird mit dem Ansatz (30) bestimmt :

$$\frac{\partial x_i}{\partial z_k} = \frac{\partial}{\partial z_k}(\mathbf{x}_i^T \mathbf{s}) = \mathbf{x}_i^T \cdot \frac{\partial \mathbf{s}}{\partial z_k} = \mathbf{x}_i^T \cdot \mathbf{s}_k \qquad k = 1,2,3$$

Für die Koordinateninkremente dx_i folgt mit (32) :

$$\begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \frac{\partial x_1}{\partial z_2} & \frac{\partial x_1}{\partial z_3} \\ \frac{\partial x_2}{\partial z_1} & \frac{\partial x_2}{\partial z_2} & \frac{\partial x_1}{\partial z_3} \end{bmatrix} * \begin{bmatrix} dz_1 \\ dz_2 \\ dz_3 \end{bmatrix}$$

$$\begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \begin{bmatrix} x_{1(1)} & x_{1(2)} & x_{1(3)} \\ x_{2(1)} & x_{2(1)} & x_{2(1)} \end{bmatrix} * \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 \end{bmatrix} * \begin{bmatrix} dz_1 \\ dz_2 \\ dz_3 \end{bmatrix}$$

$$d\mathbf{x} = \mathbf{X}\,\mathbf{S}_z\,d\mathbf{z} \tag{33}$$

Unabhängige normalisierte Ableitungen : Aus der Abhängigkeit $z_1 + z_2 + z_3 = 1$ der normalisierten Koordinaten folgt die Abhängigkeit $dz_1 + dz_2 + dz_3 = 0$ der Koordinateninkremente. Im folgenden werden z_1 und z_2 als unabhängige Variable behandelt. Daher wird $dz_3 = -dz_1 - dz_2$ in (33) substituiert. Für den vorliegenden Sonderfall $\mathbf{S}_z = \mathbf{I}$ folgt dann :

$$d\mathbf{x} = \mathbf{X}_{zu}\, d\mathbf{z}_u \tag{34}$$

$d\mathbf{z}_u$ Inkrement der unabhängigen normalisierten Koordinaten

$$\begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \begin{bmatrix} x_{1(1)} - x_{1(3)} & x_{1(2)} - x_{1(3)} \\ x_{2(1)} - x_{2(3)} & x_{2(2)} - x_{2(3)} \end{bmatrix} * \begin{bmatrix} dz_1 \\ dz_2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} * \begin{bmatrix} dz_1 \\ dz_2 \end{bmatrix}$$

Globale Ableitungen der normaliserten Koordinaten : Unter der Voraussetzung, daß die Fläche des globalen Dreiecks nicht null ist, besitzt $\mathbf{X}_{zu}$ eine Inverse $\mathbf{X}_{zu}^{-1}$, die mit $\mathbf{Z}_{xu}$ bezeichnet wird. Im Sonderfall des linearen Dreiecks kann $\mathbf{Z}_{xu}$ explizit bestimmt werden. Im allgemeinen Fall wird $\mathbf{X}_{zu}$ numerisch invertiert.

$$d\mathbf{z}_u = \mathbf{Z}_{xu}\, d\mathbf{x} \tag{35}$$

$$\mathbf{Z}_{xu} = \frac{1}{c_{11}c_{22} - c_{12}c_{21}} \begin{bmatrix} c_{22} & -c_{12} \\ -c_{21} & c_{11} \end{bmatrix}$$

Das Inkrement in z_3 wird über die Beziehung $dz_3 = -dz_1 - dz_2$ bestimmt. Dann folgt aus (35) :

$$d\mathbf{z} = \mathbf{Z}_x\, d\mathbf{x} \tag{36}$$

$$\begin{bmatrix} dz_1 \\ dz_2 \\ dz_3 \end{bmatrix} = \frac{1}{c_{11}c_{22} - c_{12}c_{21}} \begin{bmatrix} c_{22} & -c_{12} \\ -c_{21} & c_{11} \\ c_{21} - c_{22} & c_{12} - c_{11} \end{bmatrix} * \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix}$$

Globale Ableitungen des Formvektors : Die Ableitung des Formvektors **s** nach der globalen Koordinate x_i wird mit der Kettenregel bestimmt :

$$\frac{\partial \mathbf{s}}{\partial x_i} = \frac{\partial \mathbf{s}}{\partial z_1}\frac{\partial z_1}{\partial x_i} + \frac{\partial \mathbf{s}}{\partial z_2}\frac{\partial z_2}{\partial x_i} + \frac{\partial \mathbf{s}}{\partial z_3}\frac{\partial z_3}{\partial x_i}$$

Die Ableitung von **s** nach x_i wird mit $\mathbf{g}_i$ bezeichnet. Die Vektoren $\mathbf{g}_i$ und $\mathbf{g}_2$ werden spaltenweise in einer Matrix $\mathbf{S}_x$ angeordnet :

$$\mathbf{S}_x = \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 \end{bmatrix} * \begin{bmatrix} \frac{\partial \mathbf{z}}{\partial x_1} & \frac{\partial \mathbf{z}}{\partial x_2} \end{bmatrix}$$

$$\mathbf{S}_x = \mathbf{S}_z \mathbf{Z}_x \qquad (37)$$

$$\mathbf{g}_i = \frac{\partial \mathbf{s}}{\partial x_i}$$

Für den linearen Ansatz (30) im Dreieck gilt $\mathbf{S}_z = \mathbf{I}$. Aus (36) folgt :

$$\begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 \end{bmatrix} = \frac{1}{c_{11}c_{22} - c_{12}c_{21}} \begin{bmatrix} c_{22} & -c_{12} \\ -c_{21} & c_{11} \\ c_{21} - c_{22} & c_{12} - c_{11} \end{bmatrix} \qquad (38)$$

5.3 FINITER PHYSIKALISCHER ZUSTAND

5.3.1 Beschreibung des Systems

Stützstellen : Zur Beschreibung des physikalischen Zustandes des Körpers werden die Werte der Temperatur und des Wärmestroms an bestimmten Punkten des Körpers als Variable eingeführt. Diese Punkte heißen die Stützstellen der Zustandsbeschreibung. Im allgemeinen werden die Knoten des geometrischen Elementnetzes als Stützstellen gewählt.

Stützwerte : Die Werte der Temperatur und des Wäremstroms an den Stützstellen heißen Stützwerte der Zustandsbeschreibung. Auch die Ableitungen der Temperatur und des Wärmestroms können als Stützwerte eingeführt werden. Im Gegensatz zur geometrischen Interpolation zwischen Knoten mit bekannten Koordinaten sind die Stützwerte der physikalischen Interpolation zunächst teilweise unbekannt :

– An den inneren Stützstellen ist die Temperatur unbekannt.

– An den Randstützstellen ist entweder die Temperatur oder der Wäremstrom eingeprägt, die andere Zustandsgröße ist unbekannt.

Interpolation : Zwischen den Stützstellen wird der Verlauf der Temperatur und des Wärmestroms interpoliert. Dazu werden das Innere und der Rand des Körpers lückenlos in Elemente zerlegt. Die Interpolationsformeln werden elementweise als Funktionen der normalisierten Koordinaten aufgestellt. Im Inneren wird nur die Temperatur interpoliert. Auf dem Rand werden Temperatur und Wärmestrom interpoliert. Dies ist im folgenden für die ebene Scheibe gezeigt.

Primaler Systemvektor : Die Stützwerte für die Temperatur werden fortlaufend numeriert und in einem Vektor angeordnet. Dieser Vektor heißt primaler Systemvektor (Systemtemperaturvektor) und wird mit $\mathbf{u}_S$ bezeichnet. Die Reihenfolge der Stützwerte in $\mathbf{u}_S$ ist zwar beliebig, sie beeinflußt jedoch den Lösungsaufwand für die Systemgleichungen.

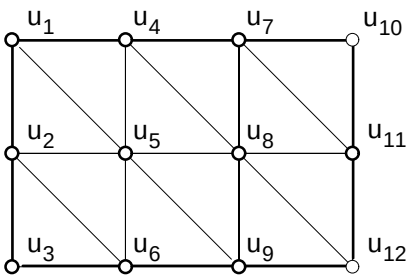
$\mathbf{u}_S$ primaler Systemvektor

Dualer Systemvektor : Die Stützwerte für den Wärmestrom an den Stützstellen in der Oberfläche des Körpers werden fortlaufend numeriert und in einem Vektor angeordnet. Dieser Vektor heißt dualer Systemvektor (Systemstromvektor) und wird mit $\mathbf{q}_S$ bezeichnet.

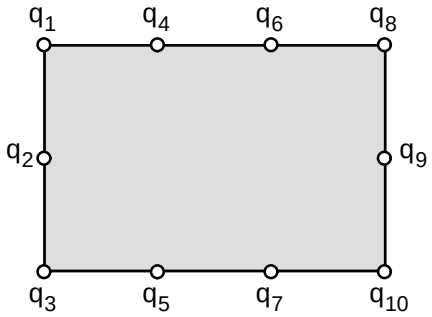
$\mathbf{q}_S$ dualer Systemvektor

Beispiel 1 : Systemvektoren für eine Scheibe

Als Stützstellen werden die Knoten des geometrischen Netzes gewählt, als Stützwerte die Temperaturen u_i an den Knoten und die Wärmeströme q_j an den Randknoten.



Temperatur



Wärmestrom

primaler Systemvektor : $\mathbf{u}_s^T$ =

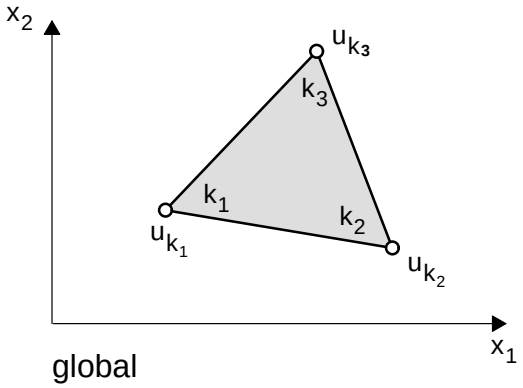
u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}

dualer Systemvektor : $\mathbf{q}_s^T$ =

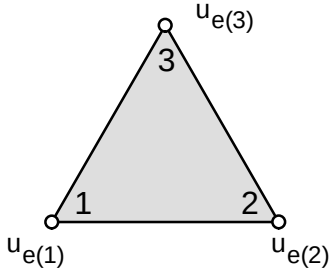
q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}

5.3.2 Interpolation der Temperatur im Inneren

Ansatz : Als Stützstellen des Elementansatzes werden die Eckpunkte eines Dreiecks gewählt. Die Eckpunkte werden lokal gegen den Uhrzeigersinn mit 1,2,3 numeriert (siehe Abschnitt 5.2). Die Stützstelle mit der lokalen Nummer i besitzt die globale Nummer k_i. Der Temperaturstützwert an dieser Stelle wird lokal mit $u_{e(i)}$ und global mit u_{ki} bezeichnet. Die Temperatur an einem beliebigen Punkt (z_1, z_2, z_3) des Dreiecks wird mit u_e bezeichnet und durch Interpolation mit einem Formvektor $\mathbf{s}_e$ bestimmt. Außerhalb des Elementes e ist der Formvektor $\mathbf{s}_e$ gleich null.



global



normalisiert

$$u_e = \mathbf{s}_e^T \mathbf{u}_e \qquad \text{für} \qquad 0 \le z_i \le 1 \tag{39}$$

$$u_e = 0 \qquad \text{außerhalb des Dreiecks e}$$

$$u_e = \begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix} \begin{bmatrix} u_{e(1)} \\ u_{e(2)} \\ u_{e(3)} \end{bmatrix}$$

$\mathbf{s}_e$ Temperaturformvektor für Element e

$\mathbf{u}_e$ Temperaturstützvektor für Element e

Topologie : Ein Stützwert u_i des primalen Systemvektors $\mathbf{u}_s$ ist im allgemeinen in den Temperaturstützvektoren $\mathbf{u}_e$ mehrerer Elemente enthalten. Der Zusammenhang wird elementweise beschrieben. Jedem Stützwert $u_{e(i)}$ des Elementvektors $\mathbf{u}_e$ entspricht ein Stützwert u_{k_i} des Systemvektors $\mathbf{u}_s$. Die Koeffizienten des Gleichungssystems werden in der Topologiematrix $\mathbf{R}_e$ des Elementzustandes zusammengefaßt.

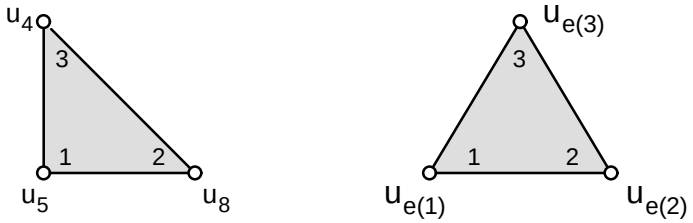
$$u_{e(i)} = u_{k_i} \qquad i \in \{1,2,3\}$$

$$\mathbf{u}_e = \mathbf{R}_e \, \mathbf{u}_s \qquad (40)$$

$\mathbf{R}_e$ Topologiematrix des Elementes e

Beispiel 2 : Topologiematrix eines Elementes aus Beispiel 1

Das Elementnetz in Beispiel 1 enthält ein Dreieck, dessen Stützwerte u_4, u_5, u_8 sind. Das folgende Diagramm zeigt die lokale Numerierung der Knoten und die Bezeichnungen $u_{e(i)}$ dieser Stützwerte im normalisierten System.



global normalisiert

$$\begin{bmatrix} u_{e(1)} \\ u_{e(2)} \\ u_{e(3)} \end{bmatrix} = \begin{array}{c} \begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} * \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{11} \\ u_{12} \end{bmatrix}$$

$\mathbf{u}_e$ $\mathbf{R}_e$ $\mathbf{u}_s$

Temperaturgradient : Der Temperaturgradient $\mathbf{g}_e$ am Punkt (z_1, z_2, z_3) eines Elementes e enthält nach der Definition (4) die Ableitungen der Temperatur u nach den globalen Variablen x_i. Bei der Ableitung des Temperaturansatzes (39) nach x_i ist der Temperaturstützvektor konstant. Die Ableitungen des Formvektors wurden in (37) bestimmt.

$$\frac{\partial u_e}{\partial x_i} = \frac{\partial}{\partial x_i}(\mathbf{s}_e^T \mathbf{u}_e) = \mathbf{g}_{ei}^T \mathbf{u}_e$$

$$\mathbf{g}_e = \mathbf{S}_{xe}^T \mathbf{u}_e \qquad (41)$$

$$\mathbf{g}_e = \begin{bmatrix} \frac{\partial u_e}{\partial x_1} \\ \frac{\partial u_e}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{e1}^T \mathbf{u}_e \\ \mathbf{g}_{e2}^T \mathbf{u}_e \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{e1}^T \\ \mathbf{g}_{e2}^T \end{bmatrix} * \begin{bmatrix} \mathbf{u}_e \end{bmatrix}$$

$$\mathbf{S}_{xe} = \begin{bmatrix} \mathbf{g}_{e1} & \mathbf{g}_{e2} \end{bmatrix}$$

Temperaturansatz im Körper : Der Temperaturansatz u_e für Element e in (39) besitzt außerhalb des Elementes e den Wert 0. Daher ist der Temperaturansatz für den Körper gleich der Summe der Ansätze für die Elemente des Körpers :

$$u_s = \sum_e \mathbf{s}_e^T \mathbf{u}_e = \sum_e \mathbf{s}_e^T \mathbf{R}_e \mathbf{u}_s \qquad (42)$$

Der Gradient $\mathbf{g}_s$ der Temperatur im Körper wird durch Ableitung des Ansatzes (42) nach den globalen Koordinaten x_i bestimmt. Analog zu (41) folgt :

$$\mathbf{g}_s = \sum_e \mathbf{S}_{xe}^T \mathbf{u}_e = \sum_e \mathbf{S}_{xe}^T \mathbf{R}_e \mathbf{u}_s \qquad (43)$$

Variation des Temperaturansatzes : Als Variationen des Temperaturverlaufs im Körper werden die Ableitungen des Ansatzes (42) nach den Stützvektoren $u_i, ..., u_n$ im primalen Systemvektor $\mathbf{u}_s$ gewählt (Methode von Galerkin) :

$$\delta u_{si} = \frac{\partial \mathbf{u}_s}{\partial u_i} = \sum_e \mathbf{s}_e^T \mathbf{R}_e \frac{\partial \mathbf{u}_s}{\partial u_i}$$

$$\delta u_{si} = \sum_e \mathbf{s}_e^T \mathbf{R}_e \mathbf{e}_i \qquad i \in \{1,2,...,n\} \qquad (44)$$

$\mathbf{e}_i$ Einsvektor mit Element 1 in Zeile i

n Anzahl der Stützwerte für Temperatur

Die Variation des Temperaturverlaufs führt zu einer entsprechenden Variation des Verlaufs des Temperaturgradienten. Diese Variation wird durch Ableitung des Ansatzes (43) nach den Stützwerten $u_i, \ldots, u_n$ bestimmt :

$$\delta \mathbf{g}_{si} = \frac{\partial \mathbf{g}_s}{\partial u_i} = \sum_e \mathbf{S}_{xe}^T \mathbf{R}_e \frac{\partial \mathbf{u}_s}{\partial u_i}$$

$$\delta \mathbf{g}_{si} = \sum_e \mathbf{S}_{xe}^T \mathbf{R}_e \mathbf{e}_i \qquad i \in \{1,2,...,n\} \qquad (45)$$

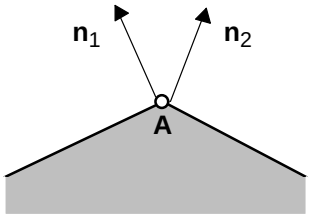
Der Wert des Temperaturansatzes u_e für Element e ist außerhalb des Elementes e gleich null. Folglich sind die Variation δu_{ei} der Temperatur und die Variation $\delta \mathbf{g}_{ei}$ des Temperaturgradienten im Element e nur vom Ansatz in diesem Element abhängig :

$$\delta u_{ei} = \mathbf{s}_e^T \mathbf{R}_e \mathbf{e}_i \qquad (46)$$

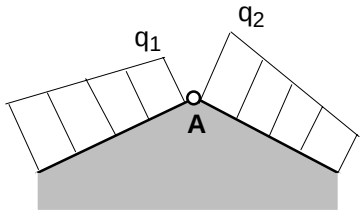
$$\delta \mathbf{g}_{ei} = \mathbf{S}_{xe}^T \mathbf{R}_e \mathbf{e}_i \qquad (47)$$

5.3.3 Interpolation auf dem Rand

Einführung : An jedem Punkt **x** der Oberfläche des Körpers gilt die Beziehung $q = \mathbf{f}^T\mathbf{n}$ zwischen dem Wärmestrom q, dem Wärmezustand **f** und der Außennormale **n**. Die Außennormale ist in der Regel an den Randknoten unstetig. Beispielsweise wird der Rand einer Scheibe durch einen Polygonzug approximiert, der an einem Teil der Randknoten geknickt ist. Folglich ist der Verlauf des Wärmestroms q auf der Oberfläche unstetig. Er kann nicht durch einen Stützwert je Stützstelle approximiert werden.



Knick im Rand
$\mathbf{n}_1 \neq \mathbf{n}_2$



Unstetiger Wärmestrom
$q_1 \neq q_2$

Approximation : Der Rand des Körpers sei stückweise gerade. Dann ist der Wärmestrom auf jeder der Kanten stetig. Er wird durch zwei konzentrierte Wärmeströme an den Endpunkten der Kante ersetzt. Die konzentrierten Wärmeströme sind Skalare und können folglich für die an den Knoten angrenzenden Kanten addiert werden. In der Integralform (29) werden also die Integrale über die Randkanten durch Summen an den Knoten ersetzt :

$$\int_{R_1} \delta u\, q\, dR \;+\; \int_{R_2} \delta u\, q_0\, dR \;=\; \sum_{K_1} \delta u_i\, Q_i \;+\; \sum_{K_2} \delta u_m Q_{mo} \qquad (53)$$

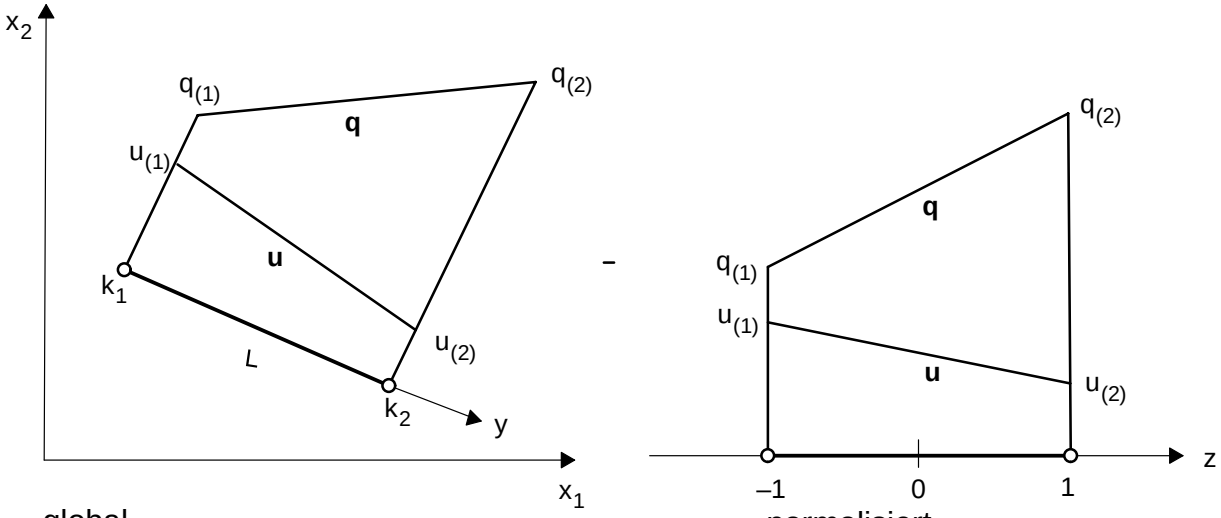
K_1 Menge der Knoten mit eingeprägter Temperatur u_{io}

K_2 Menge der Knoten mit eingeprägtem konzentriertem Wärmestrom Q_{mo}

δu_i Variation der Temperatur am Knoten i

Q_i Konzentrierter Wärmestrom am Knoten i

Ansatz : Für eine typische Randkante soll der Zusammenhang zwischen dem verteilten Wärmestrom auf der Kante und den konzentrierten Wärmeströmen an den Knoten gezeigt werden. Dazu werden zunächst die Temperatur und der Wärmefluß auf der Kante linear interpoliert. Dann werden die Integrale in (53) bestimmt und mit der rechten Seite verglichen.



global normalisiert

$$yMinimal = \frac{L}{2}(1+z) \qquad dyMinimal = \frac{L}{2}\,dz$$

$$u_r = \mathbf{s}_r^T \mathbf{u}_r = \mathbf{s}_r^T \mathbf{R}_r \mathbf{u}_s \qquad \delta u_r = \frac{\partial u_r}{\partial u_i} = \mathbf{s}_r^T \mathbf{R}_r \mathbf{e}_i \qquad (54)$$

$$\boxed{u_r} = \boxed{\tfrac{1}{2}(1-z) \mid \tfrac{1}{2}(1+z)} \boxed{\begin{matrix} u_{(1)} \\ u_{(2)} \end{matrix}}$$

$$q_r = \mathbf{s}_r^T \mathbf{q}_r \qquad (55)$$

$$\boxed{q_r} = \boxed{\tfrac{1}{2}(1-z) \mid \tfrac{1}{2}(1+z)} \boxed{\begin{matrix} q_{(1)} \\ q_{(2)} \end{matrix}}$$

Die Substitution von u_r und q_r in die linke Seite von (53) führt zu

$$\int_0^L \delta u_r \, q_r \, dy \text{Minimal} = \int_{-1}^{1} \mathbf{e}_i^T \, \mathbf{R}_r^T \, \mathbf{s}_r \, \mathbf{s}_r^T \, \mathbf{q}_r \, \frac{L}{2} \, dz$$

$$= \mathbf{e}_i^T \mathbf{R}_r^T \left(\frac{L}{8} \int_{-1}^{1} \begin{bmatrix} (1-z)^2 & 1-z^2 \\ 1-z^2 & (1-z)^2 \end{bmatrix} dz \right) \mathbf{q}_r$$

$$= \mathbf{e}_i^T \, \mathbf{R}_r^T \, \frac{L}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{q}_r \qquad (56)$$

Variiert man nacheinander die Temperatur an den Knoten k_1 und k_2, so folgt aus der Approximation (53) mit (56) :

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \frac{L}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} q_{(1)} \\ q_{(2)} \end{bmatrix} \qquad (57)$$

Verlauf des Wärmestroms auf der Oberfläche : Mit der Formel (57) kann für einen gegebenen Verlauf des eingeprägten Wärmestroms q_0 der eingeprägte konzentrierte Wärmestrom Q_0 an den Knoten berechnet werden. Für einen Knoten mit eingeprägter Temperatur u_0 kann jedoch aus dem berechneten konzentrierten Wärmestrom Q nicht auf den Verlauf des Wärmestroms auf den angrenden Rändern geschlossen werden. Dies ist die Folge der gewählten Approximation (53).

5.3.4 Finite Systemgleichungen

Substitution der Ansätze : Die Approximation (53) wird in die Integralgleichung (29) für den stationären Wärmefluß in einer Scheibe eingesetzt :

$$\int_A \delta \mathbf{g}^T \mathbf{C}\, \mathbf{g}\, dA \quad = \quad \int_A \delta u\ w_0\, dA \ - \sum_{K_1} \delta u_i\, Q_i \ - \sum_{K_2} \delta u_i\, Q_{i0}$$

Diese Gleichung soll nach der Methode von Galerkin für die Variationen δu_i und $\delta \mathbf{g}_i$ befriedigt werden, die durch Ableitung des Temperaturansatzes nach den Stützwerten $u_1, ..., u_n$ entstehen. Durch Substitution der Ansätze (43), (44) und (45) folgt für die Variation des Stützwertes u_i :

$$\int_A \{ \sum_e \mathbf{e}_i^T\, \mathbf{R}_e^T\, \mathbf{S}_{xe} \}\, \mathbf{C}\, \{ \sum_e \mathbf{S}_{xe}^T\, \mathbf{R}_e\, \mathbf{u}_s \}\, dA \quad = \int_A \{ \sum_e \mathbf{e}_i^T\, \mathbf{R}_e^T\, \mathbf{s}_e \}\, w_0\, dA$$

$$+ \text{Randbeitrag } r_i$$

Ist u_i ein Stützwert an einem Innenknoten, so ist der Randbeitrag null. Ist u_i ein Stützwert an einem Randknoten, so ist der Randbeitrag ein unbekannter konzentrierter Wärmestrom Q_i für $i \in K_1$ oder ein bekannter konzentrierter Wärmestrom Q_{io} für $i \in K_2$.

Da jeder Elementansatz und seine Variation außerhalb des Elementes gleich null sind, können die beiden Summen im Integral auf der linken Seite der Gleichung zu einer Summe zusammengefaßt werden :

$$\int_A \{ \sum_e \mathbf{e}_i^T\, \mathbf{R}_e^T\, \mathbf{S}_{xe} \}\, \mathbf{C}\, \{ \sum_e \mathbf{S}_{xe}^T\, \mathbf{R}_e\, \mathbf{u}_s \}\, dA \quad = \int_A \sum_e \mathbf{e}_i^T\, \mathbf{R}_e^T\, \mathbf{S}_{xe}\, \mathbf{C}\, \mathbf{S}_{xe}^T\, \mathbf{R}_e \mathbf{u}_s\, dA$$

Elementweise Integration : Da der gewählte Temperaturansatz (42) C_0 – Kontinuität besitzt, kann das Integral über die Fläche A ersetzt werden durch die Summe der Integrale über die Elementflächen A_e . Gleichzeitig werden Faktoren, die im Element konstant sind, aus dem Integral herausgezogen :

$$\mathbf{e}_i^T \{ \sum_e \mathbf{R}_e^T\, \mathbf{K}_e\, \mathbf{R}_e \}\, \mathbf{u}_s \quad = \quad \mathbf{e}_i^T\, \{ \sum_e \mathbf{R}_e^T\, \mathbf{w}_e \} \ + r_i \qquad\qquad i = 1,...,n \qquad (58)$$

$$\mathbf{K}_e \quad = \quad \int_{A_e} \mathbf{S}_{xe}\, \mathbf{C}_e\, \mathbf{S}_{xe}^T\, dA \quad : \qquad \text{Elementmatrix} \qquad (59)$$

$$\mathbf{w}_e \quad = \quad \int_{A_e} \mathbf{s}_e\, w_o\, dA \qquad : \qquad \text{Elementvektor} \qquad (60)$$

r_i Randbeitrag Q_i oder Q_{io} an den Randknoten, sonst 0

Systemgleichungen : Jedes der Produkte $\mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e$ in (58) ist eine quadratische Matrix mit Dimension $n * n$. Die Summation über die Elemente e führt daher zu einer Systemmatrix $\mathbf{K}_s$ mit Dimension $n * n$. Jedes der Produkte $\mathbf{R}_e^T \mathbf{w}_e$ ist ein Vektor mit Dimension n. Die Summation über die Elemente e führt daher zu einem Systemvektor $\mathbf{w}_s$ mit Dimension n. Aus (58) folgt also :

$$\mathbf{e}_i^T \mathbf{K}_s \mathbf{u}_s = \mathbf{e}_i^T \mathbf{w}_s + r_i \qquad i = 1,...,n \qquad (61)$$

$$\mathbf{K}_s = \sum_e \mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e \quad : \quad \text{Systemmatrix} \qquad (62)$$

$$\mathbf{w}_s = \sum_e \mathbf{R}_e^T \mathbf{w}_e \quad : \quad \text{Systemvektor} \qquad (63)$$

Jede der Variationen δu_i führt zu einer Gleichung der Form (61). Die Variationen $\delta u_1, ..., \delta u_n$ führen folglich zu einem System von n Gleichungen :

$$\mathbf{K}_s \mathbf{u}_s = \mathbf{w}_s - \mathbf{q}_s \qquad (64)$$

$\mathbf{u}_s$ primaler Variablenvektor (Temperatur)

$\mathbf{q}_s$ dualer Variablenvektor (Wärmestrom)

Primaler Vektor : Der primale Vektor $\mathbf{u}_s$ enthält für jeden inneren Knoten eine unbekannte Temperatur. Für jeden Randknoten enthält $\mathbf{u}_s$ eine Temperatur, die je nach Aufgabenstellung unbekannt oder eingeprägt ist.

Dualer Vektor : An einem inneren Knoten gibt es keinen Randterm. Folglich enthält der duale Vektor $\mathbf{q}_s$ für diese Knoten den Wert 0. Für jeden Randknoten enthält $\mathbf{q}_s$ entweder einen unbekannten konzentrierten Wärmestrom Q_i (wenn die Temperatur u_i eingeprägt ist) oder einen eingeprägten Strom Q_{io} . Der Wärmestrom ist positiv in Richtung der Außennormale des Randes.

Statusvektor : In jeder Zeile des Gleichungssystems (64) enthält entweder $\mathbf{u}_s$ oder $\mathbf{q}_s$ eine unbekannte Variable, während der Wert der Variable im anderen Vektor eingeprägt ist. Die Einprägungen werden in einem Statusvektor wie folgt beschrieben :

status [i] = true : in Zeile i ist der primale Wert eingeprägt

status [i] = false : in Zeile i ist der primale Wert nicht eingeprägt

6. FINITE ELEMENT MODELL

6.1 LEISTUNGSUMFANG

6.1.1 Einführung

Ziel : Die Finite Element Theorie wird in einem Finite Element Modell implementiert. Das Modell besteht aus dreieckigen finiten Elementen mit einem Freiheitsgrad (der Temperatur) an jedem Knoten. Solche Modelle können zur Berechnung des Wärmeflusses in Scheiben mit beliebiger Form und mit beliebigen Randbedingungen benutzt werden.

Plattform : Das Finite Element Modell für den stationären Wärmefluß wird in Java implementiert. Eine Optimierung des Speicherbedarfs und der Laufzeit wird nicht angestrebt. Stattdessen werden die Gesamtstruktur der Implementierung sowie die Struktur ihrer Klassen, Datenbasis und Methoden so einfach wie möglich gehalten.

Einschränkungen : Das vorliegende Modell besitzt nicht die für praktische Anwendungen erforderliche Funktionalität. Lastfälle, Lastkombinationen, Extremalwerte von Temperatur oder Wärmestrom für verschiedene Lastkombinationen, verschiedene Versionen eines Modells oder verschiedene Modelle einer Applikation können nicht behandelt werdne. Das Modell besitzt keine graphische Oberfläche für die Spezifikation der Aufgabe und die Darstellung des Wärmeflusses. Die Fehlerkontrollen bei der Eingabe ist elementar ; die Genauigkeit der Berechnungsergebnisse wird nicht durch Anpassung des Finite Element Netzes beeinflußt. Die Berücksichtigung solcher Anforderungen ist aufwendig und liegt außerhalb des Umfangs dieser Implementierung.

6.1.2 Beschreibung der Aufgabe

Eingabe : Der Anwender spezifiziert die Aufgabe mit einer Java Methode in der Klasse Heat. Diese Methode enthält eine Liste der Objekte des Modells in beliebiger Reihenfolge. Jede Anweisung der Methode besitzt die Form model.add (new ...) und erweitert das Modell um ein neu konstruiertes Objekt. Die Klasse Heat wird mit dem üblichen Kommando java Heat ausgeführt. Die Konstruktoren der Objekte sind in folgenden Klassen enthalten :

Node : Der Identifikator des Knotens, seine globalen Koordinaten und der Index der Knotentemperatur in den Systemgleichungen werden angegeben.

Element : Der Identifikator des Elementes, die Identifikationen seiner drei Knoten und der Identifikator des Materials werden angegeben.

Material : Der Identifikator des Materials und seine Wärmeleitfähigkeit werden angegeben.

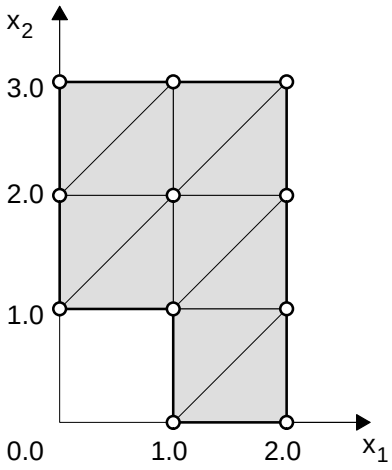
Support : Der Identifikator des Lagers und der Identifikator des Knotens, an dem das Lager angeordnet ist, sowie die am Lager eingeprägte Temperatur werden angegeben.

AreaLoad : Der Identifikator einer Flächenwärmelast und der Identifikator des Elementes, auf das die Last wirkt, sowie die Intensität des Wärmeflusses sind angegeben. Ein Wärmefluß in das Element ist positiv. Ist die Intensität über das Element konstant, so wird nur dieser konstante Wert angegeben. Variiert die Wärmelast linear im Element, so wird ihre Intensität an den drei Knoten in der Reihenfolge angegeben, in der die Knoten bei der Definition des Elementes angegeben sind.

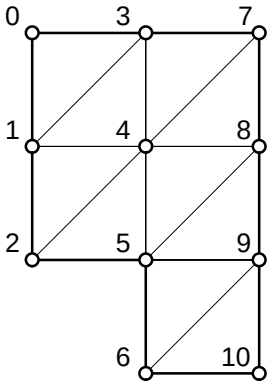
LineLoad : Eine Linienwärmelast darf nur zwischen je zwei benachbarten Knoten des Elementnetzes eingeprägt werden. Der Identifikator der Linienlast sowie die Identifikatoren der Endknoten und die Lastintensität werden angegeben. Ist die Intensität über die Linie konstant, so wird nur dieser konstante Wert angegeben. Variiert die Linienlast linear über die Strecke, so wird ihre Intensität an den Endknoten angegeben. Die Intensität ist positiv, wenn Wärme aus dem Körper abfließt.

NodeLoad : Punktwärmelasten dürfen nur an Knoten des Elementnetzes eingeprägt werden. Der Identifikator der Punktlast und der Indentifikator des Knotens, an dem die Last wirkt, sowie die Intensität der Last werden angegeben. Die Intensität ist positiv, wenn Wärme aus dem Körper abfließt.

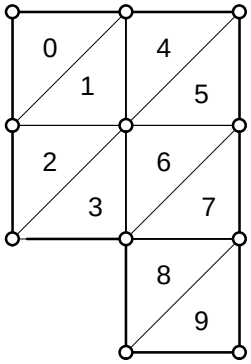
6.1.3 Rechenbeispiel



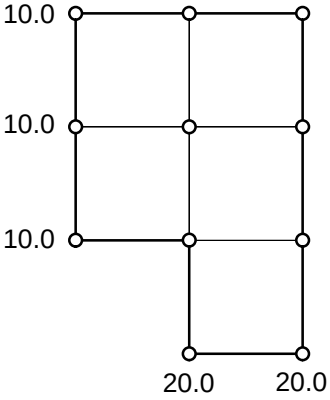
Geometrie



Knoten ”ni”



Element ”ei”



Eingeprägte Temperatur

Flächenlast	:	Elemente 2,3 w = 30.0
Material	:	c = 5.0

Der stationäre Wäremfluß in der gezeigten Scheibe wird berechnet. An einem Teil der Randknoten ist die Temperatur eingeprägt. An den anderen Randknoten ist implizit der Wärmestrom null eingeprägt. An den Elementen 2 und 3 ist ein Wärmezufluß von 30 Wärmeeinheiten je Zeit- und Flächeneinheit eingeprägt. Die Scheibe besitzt Einheitsdicke. Die folgende Klasse Heat zeigt die Eingabe für dieses Beispiel.

```
class Heat
{ static  Model    model    ;
  static  Analysis analysis ;

  public static void main(String args[])
                    throws FemException, AlgebraicException
  { setExample()       ;
    analysis.perform() ;
    model.output()     ;
  }

//..Test example.............................................

  static void setExample()
  { model    = new Model(”Skript”,”Scheibe”)                  ;
    analysis = new Analysis(”Scheibe”,”Skript”)               ;

    model.add (new Node(”n00”, 0, 0.0, 3.0))                  ;
    model.add (new Node(”n01”, 1, 0.0, 2.0))                  ;
    model.add (new Node(”n02”, 2, 0.0, 1.0))                  ;
    model.add (new Node(”n03”, 3, 1.0, 3.0))                  ;
    model.add (new Node(”n04”, 4, 1.0, 2.0))                  ;
    model.add (new Node(”n05”, 5, 1.0, 1.0))                  ;
    model.add (new Node(”n06”, 6, 1.0, 0.0))                  ;
    model.add (new Node(”n07”, 7, 2.0, 3.0))                  ;
    model.add (new Node(”n08”, 8, 2.0, 2.0))                  ;
    model.add (new Node(”n09”, 9, 2.0, 1.0))                  ;
    model.add (new Node(”n10”,10, 2.0, 0.0))                  ;

    model.add (new Element(”e0”, ”n00”, ”n01”, ”n03”, ”iso”)) ;
    model.add (new Element(”e1”, ”n01”, ”n04”, ”n03”, ”iso”)) ;
    model.add (new Element(”e2”, ”n01”, ”n02”, ”n04”, ”iso”)) ;
    model.add (new Element(”e3”, ”n02”, ”n05”, ”n04”, ”iso”)) ;
    model.add (new Element(”e4”, ”n03”, ”n04”, ”n07”, ”iso”)) ;
    model.add (new Element(”e5”, ”n04”, ”n08”, ”n07”, ”iso”)) ;
    model.add (new Element(”e6”, ”n04”, ”n05”, ”n08”, ”iso”)) ;
    model.add (new Element(”e7”, ”n05”, ”n09”, ”n08”, ”iso”)) ;
    model.add (new Element(”e8”, ”n05”, ”n06”, ”n09”, ”iso”)) ;
    model.add (new Element(”e9”, ”n06”, ”n10”, ”n09”, ”iso”)) ;

    model.add (new Material(”iso”, 5.0))                      ;

    model.add (new AreaLoad(”a2”,”e2”,30.0))                  ;
    model.add (new AreaLoad(”a3”,”e3”,30.0))                  ;

    model.add (new Support(”s00”, ”n00”, 10.0))               ;
    model.add (new Support(”s01”, ”n01”, 10.0))               ;
    model.add (new Support(”s02”, ”n02”, 10.0))               ;
    model.add (new Support(”s06”, ”n06”, 20.0))               ;
    model.add (new Support(”s10”, ”n10”, 20.0))               ;
} }
```

6.1.4 Ausgabe

Die Ausgabe der vorliegenden Implementierung ist alphanumerisch und zeilenorientiert. Die Ausgabekommandos besitzen folgende Baumstruktur :

model	:	nodes
	:	elements
	:	supports
influences	:	nodeLoads
	:	lineLoads
	:	areaLoads
behaviour	:	nodes
	:	elements
	:	supports

Die Kommandos des Zweiges “model” erzeugen Tabellen mit den vom Anwender spezifizierten Attributen der Knoten, Elemente und Lager des Modells. Die Kommandos des Zweiges “influences” erzeugen Tabellen mit den Atrributen der eingeprägten Wäremströme in Flächen, Linien und Knoten. Die Kommandos des Zweiges “behaviour” erzeugen Tabellen mit den Knotentemperaturen, dem Wärmestrom am Mittelpunkt der Elemente und dem Wärmestrom an den Lagern. Im folgenden ist die Ausgabe für das Beispiel in Abschnitt 6.1.3 gezeigt.

```
NODE COORDINATES

  Node        x1        x2

   n00      0.00      3.00
   n01      0.00      2.00
   n02      0.00      1.00
   n03      1.00      3.00
   n04      1.00      2.00
   n05      1.00      1.00
   n06      1.00      0.00
   n07      2.00      3.00
   n08      2.00      2.00
   n09      2.00      1.00
   n10      2.00      0.00

ELEMENT PROPERTIES

Element   Node1   Node2   Node3   Material

     e0     n00     n01     n03        iso
     e1     n01     n04     n03        iso
     e2     n01     n02     n04        iso
     e3     n02     n05     n04        iso
     e4     n03     n04     n07        iso
     e5     n04     n08     n07        iso
     e6     n04     n05     n08        iso
     e7     n05     n09     n08        iso
     e8     n05     n06     n09        iso
     e9     n06     n10     n09        iso
```

SUPPORT CONDITIONS

Support	Node	Temperature
s00	n00	10.000
s01	n01	10.000
s02	n02	10.000
s06	n06	20.000
s10	n10	20.000

AREA LOADS

AreaLoad	Element	Intensity		
a2	e2	30.000	30.000	30.000
a3	e3	30.000	30.000	30.000

NODAL STATE

Node	Temperature
n00	10.00
n01	10.00
n02	10.00
n03	12.56
n04	13.46
n05	15.19
n06	20.00
n07	13.33
n08	14.09
n09	16.12
n10	20.00

HEAT STATE IN ELEMENTS

Element	Heat_State_Vector	
e0	–12.82	0.00
e1	–17.32	4.50
e2	–17.32	0.00
e3	–25.97	8.66
e4	–3.83	4.50
e5	–3.15	3.83
e6	–3.15	8.66
e7	–4.63	10.13
e8	–4.63	24.03
e9	–0.00	19.40

SUPPORT BEHAVIOUR

Support	Node	Stream	Temperature
s00	n00	–6.409	10.000
s01	n01	–22.315	10.000
s02	n02	–22.987	10.000
s06	n06	12.013	20.000
s10	n10	9.698	20.000

6.2 STRUKTUR EINER APPLIKATION

6.2.1 Objekte und Klassen einer Applikation

Applikation : Die zur Behandlung einer Aufgabenmenge auf einem Rechner installierte Software heißt eine Applikation. In dieser Implementierung wird der stationäre Wärmefluß in Scheiben als eine Applikaiton behandelt.

Objekte : Die Beschreibung in Abschnitt 6.1 zeigt, daß ein Finites Element Modell aus Objekten zusammengesetzt ist. Diese Objekte sind als Knoten, Elemente, Lager, ... klassifiziert. Jede Klasse ist ein generalisierter Datentyp, der aus Attributen und Methoden zusammengesetzt ist. Die permanente Speicherung der Objekte einer Anwendung in Dateien ist nicht Bestandteil der Implementierung.

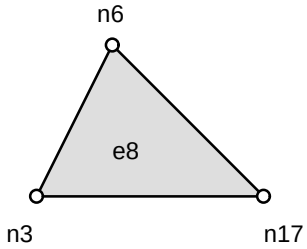
Identifikatoren : Jedes Objekt einer Applikation besitzt einen eindeutigen Identifikator. Der Identifikator ist nicht an die Klasse des Objektes gebunden. Zwei Objekte aus verschiedenen Klassen dürfen daher nicht denselben Identifikator tragen. Diese Regel erlaubt die Mengenbildung mit Objekten verschiedener Klassen, die eindeutig identifizierbar sind.

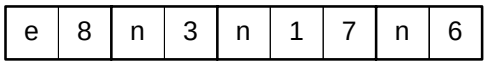
Die Applikation Wärmefluß verwendet für jedes Objekt zwei Identifikatoren. Der externe Identifikator ist ein String, den der Ingenieur festlegt. Der interne Identifikator ist eine Referenz auf das Objekt, die zur Laufzeit gebildet wird. Der externe Identifikator ist persistent : er ändert sich nicht mit der Zeit. Im Gegensatz hierzu ist der interne Identifikator temporär : er ist von der aktuellen Speicherposition des Objektes im Arbeitsspeicher oder in einer Datei abhängig.

Die Unterscheidung zwischen externen und internen Identifikatoren ist wesentlich. Mit externen Identifikatoren kann ein Objekt als Attribut eines anderen Objektes spezifiziert werden, obwohl es selbst noch nicht konstruiert ist. Beispielsweise kann der Identifikator eines Knotens als Eigenschaft eines Elementes angegeben werden, obwohl dieser Knoten noch nicht konstruiert ist. Eine vergleichbare Vorgehensweise ist mit internen Identifikatoren (Referenzen) nicht möglich : die Rferenz ist erst bekannt, nachdem das Objekt konstruiert ist.

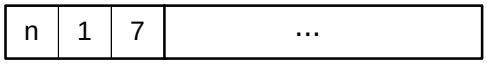
6.2.2 Beziehungen zwischen Objekten

Externe Spezifikation : Der Anwender beschreibt die Beziehungen zwischen Objekten mit ihren externen Identifikatoren. Besitzt beispielsweise ein Objekt mit dem externen Identifikator “e8” Knoten mit den Identifikatoren “n3”, “n17”, “n6”, so werden diese Identifikatoren in einem Objekt der Klasse Element gespeichert.





Objekt der Klasse Element

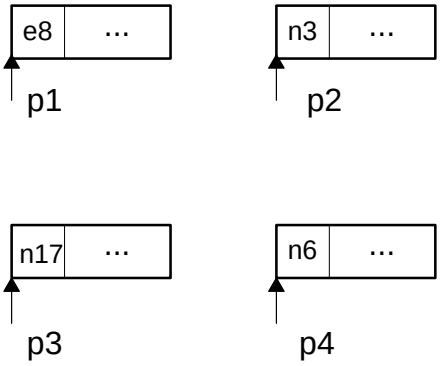


Objekt der Klasse Node

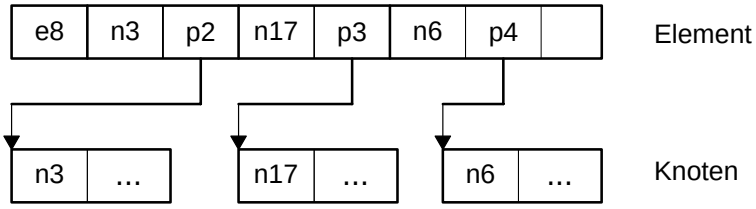
Für die interne Bearbeitung muß Element “e8” die Attribute von Objekten wie “n17” verwenden. Element “e8” muß daher die Referenz des Knotenobjektes “n17” kennen. Dies wird mit dem Objektverzeichnis der Klasse App erreicht.

Objektverzeichnis : Das Objektverzeichnis ist ein statisches Attribut der Klasse App. Der Datentyp des Objektverzeichnisses ist HashMap aus dem Paket java.util. Hashtabellen sind im Skript Datenstrukturen erläutert. Jedes Objekt der Applikation wird bei seiner Konstruktion in das Objektverzeichnis eingetragen. Die Eintragung besteht aus dem externen Identifikator des Objektes als Schlüssel und seiner Referenz als Wert. Alle Objekte, die das Finite Element Modell beschreiben, werden in das Objektverzeichnis eingetragen, ehe die internen Beziehungen zwischen den Objekten hergestellt werden. Das Objektverzeichnis des vorangehenden Beispiels besitzt dann folgenden Inhalt :

Schlüssel	Wert
e8	p1
n3	p2
n17	p3
n6	p4



Verknüpfung : Nachdem das Modell spezifiziert ist und ehe sein Verhalten berechnet werden kann, müssen die Objekte des Modells intern verknüpft werden. Zur Verknüpfung des Elementes “e8” werden die im Objekt gespeicherten Strings als Schlüssel im Objektverzeichnis verwendet um die Referenzen p2, p3, p4 der Kontenobjekte zu lesen. Diese Referenzen werden dann im Objekt “e8” gespeichert. Mit diesen Referenzen liest das Objekt “e8” die Knotenobjekte “n3”, “n17” und “n6” :



Zur Ausführung dieser Verknüpfungen enthält jede Klasse, die externe Identifikatioren als Attribute besitzt, eine Methode mit der Bereicherung setReferences ().

6.2.3 Klasse App(lication)

Inhalt : Das HashMap objectMap dieser Klasse enthält die String-Identifikatoren und die Referenzen aller Objekte der Applikation. Die anderen Klassen verwenden die Methoden der Klasse App um die Objekte im objectMap zu verwalten :

addObject (object)	:	trägt das Objekt in das Verzeichnis ein
getObject (identifier)	:	liefert die Referenz des Objektes
containsObject (identifier)	:	liefert true oder false
removeObject (identifier)	:	löscht das Objekt im Verzeichnis
numberOfObjects ()	:	liefert die Anzal der Objekte
clearObjects ()	:	löscht alle Objekte im Verzeichnis

Code :

```
public class App
{ private static  HashMap  objectMap ;
  static { objectMap = new HashMap() ; }

//......Methods for objects.................................

  public static AppObject addObject(AppObject object)
  { if (containsObject(object.id))
    { Z.putS(" *** Application already contains Object ") ;
      Z.putS(object.id + " ***")                          ;
      Z.newL()                                            ;
      return object                                       ;
    }
    return (AppObject)objectMap.put(object.id,object)     ;
  }

  public static AppObject getObject(String objectId)
  { return (AppObject)objectMap.get(objectId); }

  public static  boolean containsObject(String objectId)
  { return objectMap.containsKey(objectId); }

  public static int numberOfObjects()
  { return objectMap.size(); }

  public static AppObject removeObject(String objectId)
  { return (AppObject)objectMap.remove(objectId); }

  public static void clearObjects()
  { objectMap.clear(); }
```

```
//......Print the defined Objects.............................
  public static void printObjects()
  { Z.putS(" List of Objects in the Application ")           ;
    Z.newL()                                                 ;
    ArrayList objectList = new ArrayList(objectMap.keySet()) ;
    Collections.sort(objectList)                             ;
    ListIterator iter = objectList.listIterator()            ;
    int counter = 0                                          ;
    while(iter.hasNext())
    { Z.putS(10,(String)iter.next()) ;
      if ((counter++) == 5)
      { Z.newL()    ;
        counter = 0 ;
} } } }
```

6.2.4 Klasse AppObject

Inhalt : AppObject ist die Superklasse aller Objekt der Applikation. Der Konstruktor AppObject(identifier) trägt das Objekt in das Objektverzeichnis der Klasse App(lication) ein. Die Klasse implementiert das Interface Comparable, sodaß die Objekte der Applikation lexicographisch geordnet werden können.

Code :

```
public class AppObject implements Comparable
{ String id ;
  public AppObject(String id)
  { this.id = id        ;
    App.addObject(this) ;
  }
  public String getId() { return id; }
  public int compareTo(Object object)
  { String s = ((AppObject)object).id ;
    return id.compareTo(s)            ;
} }
```

6.3 IMPLEMENTIERUNG DER MODELLKOMPONENTEN

6.3.1 Klasse Node

Jedes Objekt dieser Klasse beschreibt einen Knoten des Modells. Die Knoten werden mit folgenden Methoden konstruiert und behandelt :

Node(identifier, systemIndex, x1, x2)
void setTemperature (value)
void set SystemIndex (value)
void set Coordinates (x1, x2)
int get SystemIndex ( )
double getTemperature ( )
double[ ] getCoordinates ( )

6.3.2 Klasse Support

Jedes Objekt dieser Klasse spezifiziert die Einprägung einer Temperatur an einen bestimmten Knoten des Modells. Die Lager werden mit folgenden Methoden konstruiert und behandelt :

Support(identifier, nodeIdentifier, temperature)
void setNodeId (nodeIdentifier)
void setTemperature (value)
void setReerences ( )
String getNodeId ( )
double getTemperature ( )

6.3.3 Klasse Element

Jedes Objekt dieser Klasse beschreibt ein Element des Modells. Die zur Berechnung der Elementeigenschaften benötigten Speicherbereiche werden als Klassenattribute definiert, die von allen Elementen der Klasse gemeinsam genutzt werden. Die Elemente werden mit folgenden Methoden konstruiert und behandelt :

Element(identifier, idNode1, idNode2, idNode3, idMaterial)
void setNodeId (idNode1, idNode2, idNode3)
void setMaterialId (idMaterial)
void setReferences( )
String[ ] getNodeId( )
Node[ ] getNodeReferences( )

String	getMaterialId( )
double[ ][ ]	getNodeCoordinates( )
int[ ]	getSystemIndices( )
double[ ]	getTemperatureVector( )
void	computeArea( )
void	computeSx( )
double[ ][ ]	computeMatrix( )
double[ ]	computeHeatState( )

6.3.4 Berechnung der Elementmatrix

In der Methode computeMatrix( ) der Klasse Element ist der Algorithmus zur Berechnung der Elementmatrix (59) implementiert. Diese Methode unterschiedet sich von allgemeinen Methoden zur Berechnung von Elementmatrizen in folgenden Punkten :

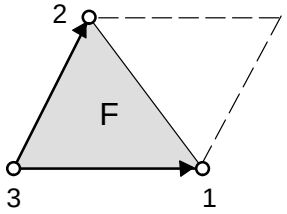
(1) Die Matrix $\mathbf{S}_z$ wird nicht explizit berechnet, da ihr Wert **I** ist.

(2) Die Matrix $\mathbf{X}_{zu}$ wird nicht numerisch invertiert, da die Inverse $\mathbf{Z}_x$ gut mit der Cramer'schen Regel bestimmt werden kann.

(3) Das Produkt $\mathbf{S}_x = \mathbf{S}_z\,\mathbf{Z}_x$ wird nicht explizit berechnet, da $\mathbf{S}_z = \mathbf{I}$.

(4) Da das Material der Scheibe isotrop ist, gilt für die Wärmeleitmatrix $\mathbf{C} = c\,\mathbf{I}$. Das Produkt $\mathbf{S}_x\,\mathbf{C}\,\mathbf{S}_x^T$ wird daher durch $c\,\mathbf{S}_x\,\mathbf{S}_x^T$ ersetzt.

(5) Die Integration über die Fläche des Dreiecks wird nicht numerisch ausgeführt, da der Integrand konstant und die analytische Integration einfach ist. Die Fläche des Dreiecks wird über das Kreuzprodukt ermittelt :



$$F = \frac{1}{2}\,\mathbf{e}_3 \cdot \{(\mathbf{x}_1 - \mathbf{x}_3) \times (\mathbf{x}_2 - \mathbf{x}_3)\}$$

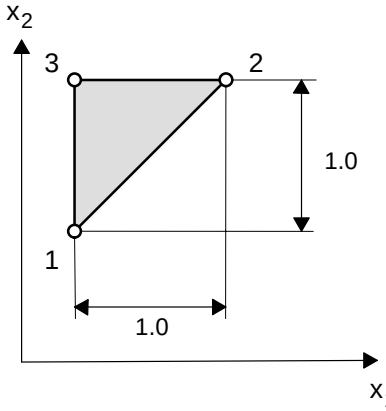
$$F = \frac{1}{2}\,(c_{11}\,c_{22} - c_{12}\,c_{21}) = \frac{1}{2}\,\det \mathbf{X}_{zu}$$

Die Methode enthält folgenden Code :

```
double[][] computeMatrix() throws FemException
{ computeArea() ;
  computeSx()   ;
  for (int i=0; i < 3; i++)              // Sx*C
  { for (int m=0; m < 2; m++)
    { tt[i][m] = material.conductivity * sx[i][m] ;
  } }
  for (int i=0; i < 3; i++)              // Ke = F*Sx*C*Sx(t)
  { for (int m=0; m < 3; m++)
    { double sum = 0.0 ;
      for (int k=0; k < 2; k++)
      { sum += tt[i][k] * sx[m][k] ;
      }
      matrix[i][m] = area * sum ;
  } }
  return matrix ;
}
```

Die folgenden Beispiele zeigen den Rechengang.

Beispiel 1 : Elementmatrix des Elementes 1



$c_{11} = 0.0 \qquad c_{12} = 1.0$

$c_{21} = -1.0 \qquad c_{22} = 0.0$

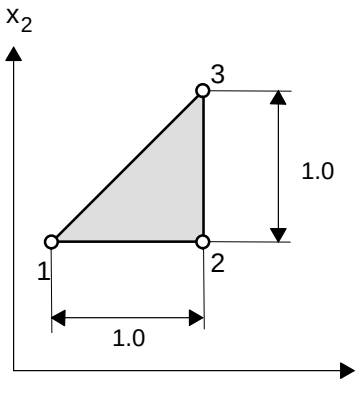
$c_{11}c_{22} - c_{12}c_{21} = 1.0$

$\text{flaeche} = 0.5$

$$\mathbf{S}_x = \begin{bmatrix} 0 & -1.0 \\ 1.0 & 0 \\ -1.0 & 1.0 \end{bmatrix} \qquad \mathbf{S}_x\,\mathbf{C}\,\mathbf{S}_x^T = 5.0\,\mathbf{S}_x\,\mathbf{S}_x^T = \begin{bmatrix} 5.0 & 0 & -5.0 \\ 0 & 5.0 & -5.0 \\ -5.0 & -5.0 & 10.0 \end{bmatrix}$$

$$\mathbf{K}_e = \begin{bmatrix} 2.5 & 0 & -2.5 \\ 0 & 2.5 & -2.5 \\ -2.5 & -2.5 & 5.0 \end{bmatrix}$$

Beispiel 2 : Elementmatrix des Elementes 2



$$c_{11} = -1.0 \qquad c_{12} = 0.0$$
$$c_{21} = -1.0 \qquad c_{22} = 1.0$$
$$c_{11}c_{22} - c_{12}c_{21} = 1.0$$
$$\text{flaeche} = 0.5$$

$$\mathbf{S}_x = \begin{bmatrix} -1.0 & 0 \\ 1.0 & -1.0 \\ 0 & 1.0 \end{bmatrix} \qquad \mathbf{S}_x \mathbf{C} \mathbf{S}_x^T = 5.0\, \mathbf{S}_x \mathbf{S}_x^T = \begin{bmatrix} 5.0 & -5.0 & 0 \\ -5.0 & 10.0 & -5.0 \\ 0 & -5.0 & 10.0 \end{bmatrix}$$

$$\mathbf{K}_e = \begin{bmatrix} 2.5 & -2.5 & 0 \\ -2.5 & 5.0 & -2.5 \\ 0 & -2.5 & 2.5 \end{bmatrix}$$

6.3.5 Klasse Material

Jedes Objekt dieser Klasse beschreibt ein Material des Modells. Dieses Material kann mehreren Elementen des Modells zugeordnet werden. Die Materialien werden mit folgenden Methoden konstruiert und behandelt :

Material (identifier, conductivity)
double getConductivity( )

6.3.6 Klasse NodeLoad

Jedes Objekt dieser Klasse beschreibt die Einprägung eines Wärmestroms an einem Knoten des Modells. Der Wärmestrom ist positiv, wenn Wärme aus dem Körper abfließt. Knotenlasten werden mit folgenden Methoden konstruiert und behandelt :

NodeLoad (identifier, nodeIdentifier, stream)
void setReferences( )
String getNodeId( )
Node getNodeReference( )
double getStream( )

6.3.7 Klasse LineLoad

Jedes Objekt dieser Klase beschreibt die Einprägung eines Wärmestroms auf einer Kante des Modells. Der Wärmestrom ist positiv, wenn Wärme aus dem Körper abfließt. Linienlasten werden mit folgenden Methoden konstruiert und behandelt :

LineLoad(identifier, idNode1, idNode2, intensity)
LineLoad(identifier, idNode1, idNode2, value1, value2)
void setNodeId(idNode1, idNode2)
void setIntensity(value1, value2)
void setReferences( )
String[ ] getNodeId( )
Node[ ] getNodeReferences( )
double[ ] getIntensity( )
double[ ] computeVector( )

6.3.8 Klasse AreaLoad

Jedes Objekt dieser Klasse beschreibt die Einprägung eines Wärmestroms in einem Element des Modells. Der Wärmestrom ist positiv, wenn Wärme in den Körper hineinfließt. Flächenlasten werden mit folgenden Methoden konstruiert und behandelt :

AreaLoad(identifier, elementIdentifier, intensity)
void setIntensity (value1, value2, value3)
void setReferences( )
String getElementId( )
Element getElementReference( )
int[ ] getSystemIndices( )
double[ ] getIntensity( )
double[ ] computeLoadVector( )

6.3.9 Berechnung des Elementvektors

Die Methode computeVector( ) unterscheidet sich von allgemeinen Methoden zur Berechnung von Elementvektoren durch die analytische Integration über die Fläche des Dreiecks. Dem Inhalt $|\,\mathbf{e}_1 \times \mathbf{e}_2\,|$ einer Einheitsfläche im globalen System entspricht der Inhalt $|\,\mathbf{j}_1 \times \mathbf{j}_2\,|$ des Kreuzproduktes der Basisvektoren im normalisierten System :

$$\mathbf{j}_1 = \frac{\partial \mathbf{x}}{\partial z_1} = \begin{bmatrix} \frac{\partial x_1}{\partial z_1} \\ \frac{\partial x_2}{\partial z_1} \end{bmatrix} \qquad \mathbf{j}_2 = \frac{\partial \mathbf{x}}{\partial z_2} = \begin{bmatrix} \frac{\partial x_1}{\partial z_2} \\ \frac{\partial x_2}{\partial z_2} \end{bmatrix}$$

$$|\,\mathbf{j}_1 \times \mathbf{j}_2\,| = \det \mathbf{X}_{zu}$$

$$\mathbf{w}_e = \int_{A_e} \mathbf{s}_e\, w_0\, dA_e = \int_{A_n} \mathbf{s}_e\, w_0 \det \mathbf{X}_{zu}\, dz_1\, dz_2$$

$$\mathbf{w}_e = w_0 \det \mathbf{X}_{zu} \int_{A_n} \mathbf{s}_e\, dz_1\, dz_2 \qquad (65)$$

A_e Fläche des Dreiecks im globalen System

A_n Fläche des Dreiecks im normalisierten System

Für die Integration im normalisierten Koordinatensystem gilt :

$$\det \mathbf{X}_{zu} = 2F$$

$$\int_{A_n} z_1\, dz_1\, dz_2 = \int_0^1 \left\{ \int_0^{1-z_1} dz_2 \right\} z_1\, dz_1 = \int_0^1 z_1 (1 - z_1)\, dz_1 = \frac{1}{6}$$

$$\int_{A_n} z_2\, dz_1\, dz_2 = \int_0^1 \left\{ \int_0^{1-z_1} z_2\, dz_2 \right\} dz_1 = \int_0^1 \frac{1}{2} (1 - z_1)^2\, dz_1 = \frac{1}{6}$$

$$\int_{A_n} z_3\, dz_1\, dz_2 = \int_0^1 \left\{ \int_0^{1-z_1} (1 - z_1 - z_2)\, dz_2 \right\} dz_1 = \int_0^1 \frac{1}{2} (1 - z_1)^2\, dz_1 = \frac{1}{6}$$

$$\mathbf{w}_e = \frac{w_0 F}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad (66)$$

F Fläche des Dreiecks im globalen System

w_0 Wärmequelle im Element

Die Methode computeLoadVector( ) enthält den Algorithmus zur Berechnung des Elementvektors :

```
public double[] computeLoadVector()
{ Element element                                    ;
  int     point,i                                    ;
  double  factor,coor[]                              ;

  element = (Element)App.getObject(elementId)        ;
  element.computeArea()                              ;
  for (point = 0; point < 3; point++)
  { coor     = gaussCoordinates[point]               ;
    factor   = coor[0] * intensity[0] +
               coor[1] * intensity[1] +
               coor[2] * intensity[2]                ;
    factor  *= element.area2 * gaussWeight[point]    ;
    if (point == 0)
    { vector[0]  = factor * coor[0] ;
      vector[1]  = factor * coor[1] ;
      vector[2]  = factor * coor[2] ;
    }
    else
    { vector[0] += factor * coor[0] ;
      vector[1] += factor * coor[1] ;
      vector[2] += factor * coor[2] ;
  } }
  return vector ;
}
```

6.4 IMPLEMENTIERUNG DES MODELLS

6.4.1 Einführung

Die in Abschnitt 6.3 behandelten Klassen für Komponenten des Modells werden um Klassen ergänzt, in denen die Komponenten als Mengen verwaltet und die zur Berechnung des stationären Wärmeflusses entwickelten Algorithmen ausgeführt werden :

Model : Mit einem Objekt dieser Klasse werden die Objekte eines Finite Element Modells für stationären Wärmefluß verwaltet.

Equation : In einem Objekt dieser Klasse wird ein Gleichungssystem der Form (64) mit Profilstruktur der Koeffizientenmatrix gespeichert. Die Klasse enthält Objektmethoden zum Aufbau und zur Lösung des Gleichungssystems. Diese Klasse ist unabhängig von der physikalischen Aufgabe und kann allgemein für lineare Gleichungssysteme mit Profilstruktur verwendet werden.

Analysis : Ein Objekt dieser Klasse steuert die Berechnung des Wärmestroms in einem Finite Element Modell der Klasse Model. Es verwendet dafür ein Objekt der Klasse Equation.

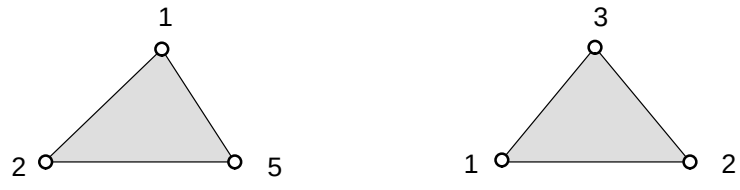
Diese drei Klassen sind in nachfolgenden Abschnitten beschrieben. Zunächst werden die Profilstruktur und die Speicherungsstruktur der Koeffizientenmatrix behandelt.

6.4.2 Profil der Systemmatrix

Einführung : Die Systemmatrix $\mathbf{K}_s = \sum \mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e$ wird in der Klasse Analysis in einer Schleife über die Elemente des Systems durch Addition der Elementbeiträge aufgebaut. In der Berechnung des Elementbeitrages $\mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e$ zur Systemmatrix $\mathbf{K}_s$ wird die Multiplikation mit der Topologiematrix $\mathbf{R}_e$ durch die Abbildung der Elementindizes auf Systemindizes ersetzt.

Die Systemmatrix besitzt Profilstruktur und ist symmetrisch. Diese Eigenschaften bleiben beim Zerlegen der Systemmatrix während des Lösens der Systemgleichungen erhalten. Daher wird die Systemmatrix zeilenweise ab dem Profil bis zum Diagonalelement gespeichert. Das Profil wird automatisch ermittelt.

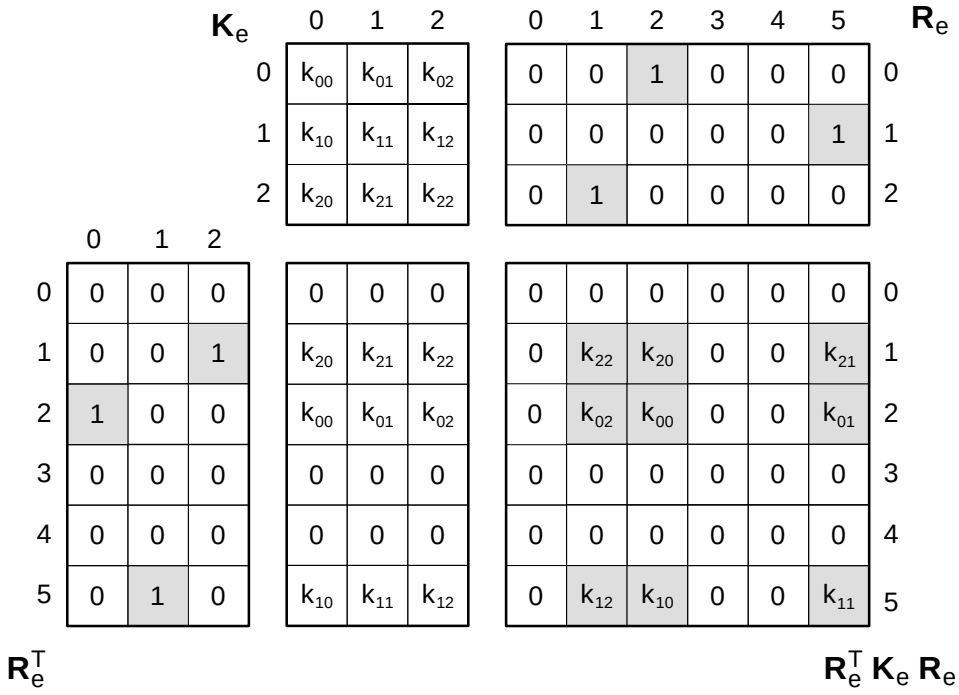
Abbildung der Indizes : Das folgende Beispiel zeigt die globalen Indizes 2,5,1 und die lokalen Indizes 1,2,3 der Zustandsvariablen an den Knoten eines Dreiecks. Das Element besitzt folgende Topologiematrix :



		0	1	2	3	4	5		
$u_{(1)}$		0	0	1	0	0	0		$u_{(0)}$
$u_{(2)}$	=	0	0	0	0	0	1	*	$u_{(1)}$
$u_{(3)}$		0	1	0	0	0	0		$u_{(2)}$
									$u_{(3)}$
									$u_{(4)}$
									$u_{(5)}$
$\mathbf{u}_e$		$\mathbf{R}_e$							$\mathbf{u}_s$

$$\mathbf{u}_e = \mathbf{R}_e \, \mathbf{u}_s$$

Die Koeffizienten der Elementmatrix $\mathbf{K}_e$ seien $\mathbf{K}_{im}$. Dann führt die explizite Multiplikation $\mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e$ zu folgendem Ergebnis :



Die Multiplikation $\mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e$ der Elementmatrix $\mathbf{K}_e$ mit der Topologiematrix $\mathbf{R}_e$ verteilt die Koeffizienten von $\mathbf{K}_e$ auf die Systemmatrix $\mathbf{K}_s$. Diese Verteilung kann durch eine Abbildung $m : \{1,2,3\} \rightarrow \{m_1, m_2, m_3\}$ mit $m(i) = m_i$ ersetzt werden. Die Abbildung m wird mit einem Indexvektor beschrieben. Der Koeffizient k_{is} von $\mathbf{K}_e$ wird zu dem Koeffizienten $k_{m_i m_s}$ von $\mathbf{K}_s$ addiert. Das vorangehende Beispiel besitzt folgenden Indexvektor:

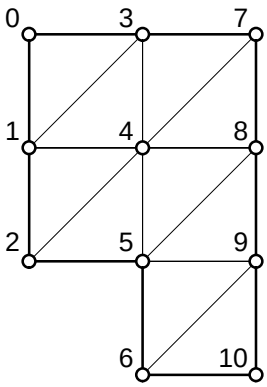
i	1	2	3
m_i	2	5	1

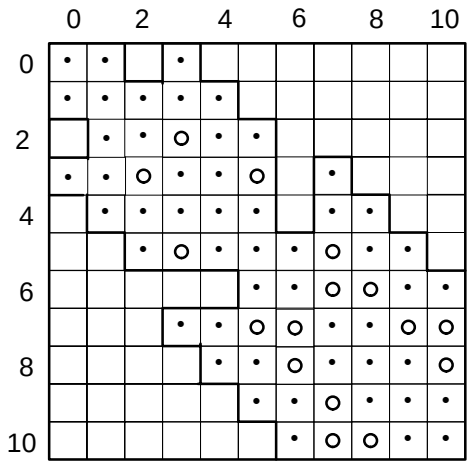
$k_{23} \in \mathbf{K}_e \rightarrow k_{m_2 m_3} = k_{51} \in \mathbf{K}_s$

Profilvektor der Systemmatrix : Bei geeigneter Wahl der Systemindizes der Knotentemperaturen u_i besitzt die Systemmatrix $\mathbf{K}_s$ eine Profilstruktur. Diese Profilstruktur wird zur Verringerung des Lösungsaufwandes für die Systemgleichungen genutzt. Das untere Profil wird mit dem Zeilenprofilvektor zProfile[ ] beschrieben, das obere Profil mit dem Spaltenprofilvektor sProfile[ ]. Wegen der Symmetrie von $\mathbf{K}_s$ sind die beiden Profile gleich und werden durch einen gemeinsamen Profilvektor profile[ ] ersetzt.

zProfile[i]		Spaltenindex des ersten Koeffizienten in Zeile i von $\mathbf{K}_s$, der nicht null ist (von links gesehen)
sProfile[i]		Zeilenindex des ersten Koeffizienten der Spalte i von $\mathbf{K}_s$, der nicht null ist (von oben gesehen)
profile[i]	= zProfile[i] =	sProfile[i]

Beispiel : Profilvektor der Systemmatrix des Rechenbeispiels 6.1.3





profile[ ]
0
0
1
0
1
2
5
3
4
5
6

Netz mit Systemindizes Systemmatrix profile[ ]

Aufbau des Profilvektors : Da die Diagonale der Systemmatrix $\mathbf{K}_s$ immer belegt ist, wird die Zeile i des Profilvektors auf den Anfangswert profil[i] = i gesetzt. Die Elemente des Netzes werden nacheinander betrachtet. Für ein allgemeines Element e wird der Indexvektor bestimmt. In einer Doppelschleife über den Indexvektor werden für jedes Element $k_{is} \in K_e$ die Systemindizes m_i und m_s gelesen. Ist das Profil von $\mathbf{K}_s$ in Zeile m_i größer als m_s, so wird profil[m_i] = m_s gesetzt.

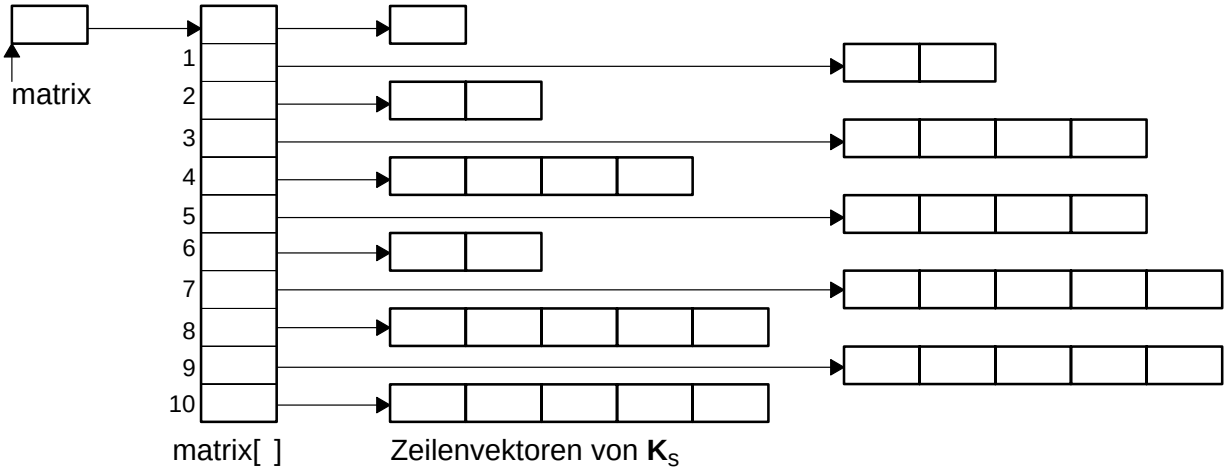
$$\bigwedge_i \bigwedge_s \left(\text{profil}[m_i] > m_s \quad \Rightarrow \quad \text{profile}[m_i] = m_s \right)$$

6.4.3 Koeffizienten der Systemmatrix

Datenstruktur : Die Systemmatrix $\mathbf{K}_S$ wird zeilenweise als Array gespeichert. Der Vektor für Zeile i enthält aufeinanderfolgend die Koeffizienten ab Spalte profile[i] bis Spalte i. Folglich enthält der Zeilenvektor i – profile[i] + 1 Koeffizienten. Da die Nullkoeffizienten am Zeilenanfang nicht gespeichert werden, kann nicht mit der üblichen Indizierung matrix[i][k] auf die Koeffizienten von $\mathbf{K}_S$ zugegriffen werden.

Die Referenz matrix und der Referenzvektor matrix[ ] für die Zeilen von $\mathbf{K}_S$ werden bereits im Konstruktor Equation( ) zugewiesen. Die Zeilenvektoren werden nach Aufstellen des Profilvektors in der Methode allocateMatrix( ) von Klasse Equation zugewiesen.

Beispiel : Systemmatrix des Beispiels in Abschnitt 6.4.2



Zugriff auf einen Koeffizienten von $\mathbf{K}_S$: Der Zugriff auf einen Koeffizienten von $\mathbf{K}_S$ ist nur mit privaten Methoden der Klasse Equation möglich :

double **getValue** (int row, int column)

void **putValue** (int row, int column, double value)

void **addValue** (int row, int column, double value)

Der Zugriff auf die Koeffizienten von $\mathbf{K}_S$ mit diesen Methoden wurde mit dem Ziel der guten Lesbarkeit der Methoden der Klasse Equation gewählt. Die Ausführungsgeschwindigkeit der Berechnungen kann verbessert werden, indem die Methodenaufrufe durch in-line-code mit Zeigern auf die aktuellen Zeilen der Datenstrukturen ersetzt werden.

Systemgleichungen des Rechenbeispiels

$\mathbf{K}_s =$

5.0	–2.5		–2.5								0
–2.5	10.0	–2.5	0	–5.0							1
	–2.5	5.0		0	–2.5						2
–2.5	0		10.0	–5.0			–2.5				3
	–5.0	0	–5.0	20.0	–5.0		0	–5.0			4
		–2.5		–5.0	15.0	–2.5		0	–5.0		5
					–2.5	5.0			0	–2.5	6
			–2.5	0			5.0	–2.5			7
				–5.0	0		–2.5	10.0	–2.5		8
					–5.0	0		–2.5	10.0	–2.5	9
						–2.5			–2.5	5.0	10
0	1	2	3	4	5	6	7	8	9	10	

$\mathbf{u}_s =$

10.0
10.0
10.0
u_3
u_4
u_5
20.0
u_7
u_8
u_9
20.0

$\mathbf{w}_s =$

0
5.0
10.0
0
10.0
5.0
0
0
0
0
0

$\mathbf{q}_s =$

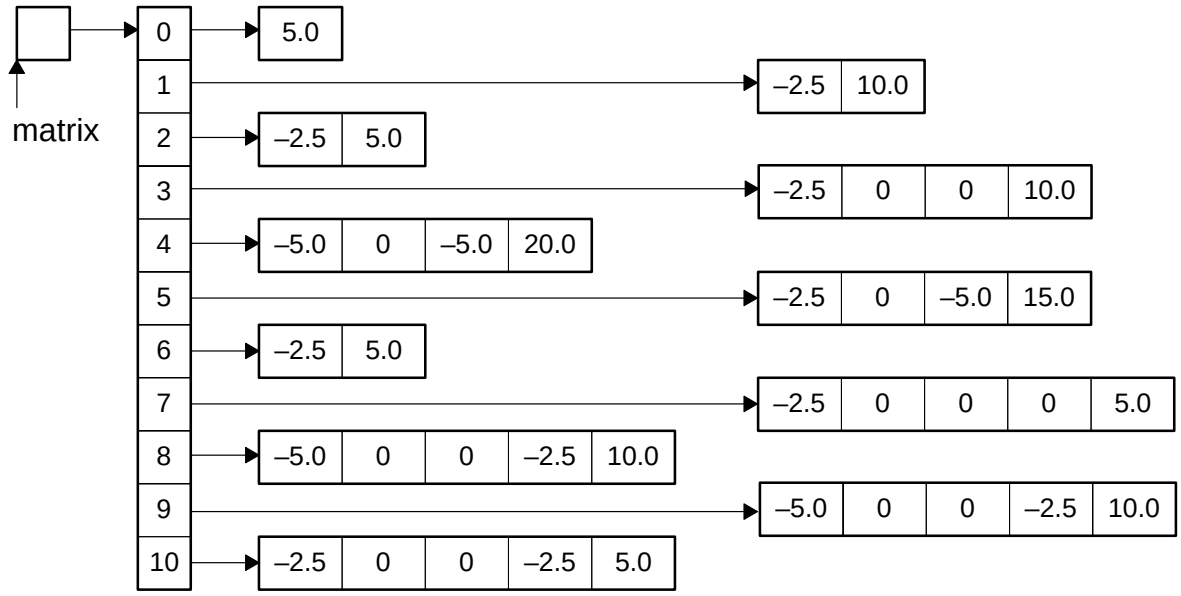
q_0
q_1
q_2
0
0
0
q_6
0
0
0
q_{10}

profile =

0
0
1
0
1
2
5
3
4
5
6

status =

true
true
true
f
f
f
true
f
f
f
true



6.4.4 Lösung mit Statusvektor

Theorie : Gegeben sei ein lineares Gleichungssystem **Ax** = **c** + **y** mit voll belegter Systemmatrix **A** und Statusvektor **s**. Die Koeffizienten von **A** und **c** seien bekannt. Besitzt s[i] den Wert true, so ist x[i] bekannt und y[i] unbekannt. Besitzt s[i] den Wert false, so ist x[i] unbekannt und y[i] bekannt. Gesucht werden die unbekannten Variablen in **x** und **y**.

Zur Lösung des Gleichungssystems wird der Vektor **x** so umgeordnet, daß er in den oberen Zeilen die unbekannten Variablen und in den unteren Zeilen die bekannten Variablen enthält. Die Matrix **A** und die Vektoren **c**, **y** werden dementsprechend umgeordnet. Dannach besitzt das lineare Gleichungssystem folgende Blockstruktur :

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} * \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}$$

$\mathbf{x}_1$, $\mathbf{y}_2$	unbekannte Variable
$\mathbf{x}_2$, $\mathbf{y}_1$	eingeprägte Variable

Das lineare Gleichungssystem zerfällt also in zwei Teilsysteme :

$$\mathbf{A}_{11}\,\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 \;=\; \mathbf{c}_1 + \mathbf{y}_1 \tag{67}$$

$$\mathbf{A}_{21}\,\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 \;=\; \mathbf{c}_2 + \mathbf{y}_2 \tag{68}$$

In das erste Teilsystem werden die Werte von $\mathbf{x}_2$ und $\mathbf{y}_1$ eingesetzt. Das resultierende Gleichungssystem wird nach $\mathbf{x}_1$ aufgelöst :

$$\mathbf{A}_{11}\,\mathbf{x}_1 = \mathbf{h}_1 \quad \text{mit} \quad \mathbf{h}_1 = \mathbf{c}_1 + \mathbf{y}_1 - \mathbf{A}_{12}\mathbf{x}_2 \tag{69}$$

$$\mathbf{x}_1 \;= \mathbf{A}_{11}^{-1}\,\mathbf{h}_1 \tag{70}$$

In das zweite Teilsystem werden die Werte von $\mathbf{x}_1$ und $\mathbf{x}_2$ eingesetzt. Das resultierende Gleichungssystem wird nach $\mathbf{y}_2$ aufgelöst :

$$\mathbf{y}_2 \;= \mathbf{A}_{21}\mathbf{x}_1 + \;\mathbf{A}_{22}\mathbf{x}_2 - \mathbf{c}_2 \tag{71}$$

Algorithmus : Die Umordnung des Gleichungssystems $\mathbf{A}\,\mathbf{x} = \mathbf{c} + \mathbf{y}$ in die Teilsysteme (67) und (68) ist im Rechner aufwendig und stört die Profilstruktur. Der mit den Gleichungen (69) bis (71) gezeigte Lösungsweg wird daher wie folgt implementiert :

– In den Zeilen mit Status false wird die Substitution (69) ausgeführt.

– Die Zeilen und Spalten mit Status false werden zerlegt : $\mathbf{A}_{11} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$.

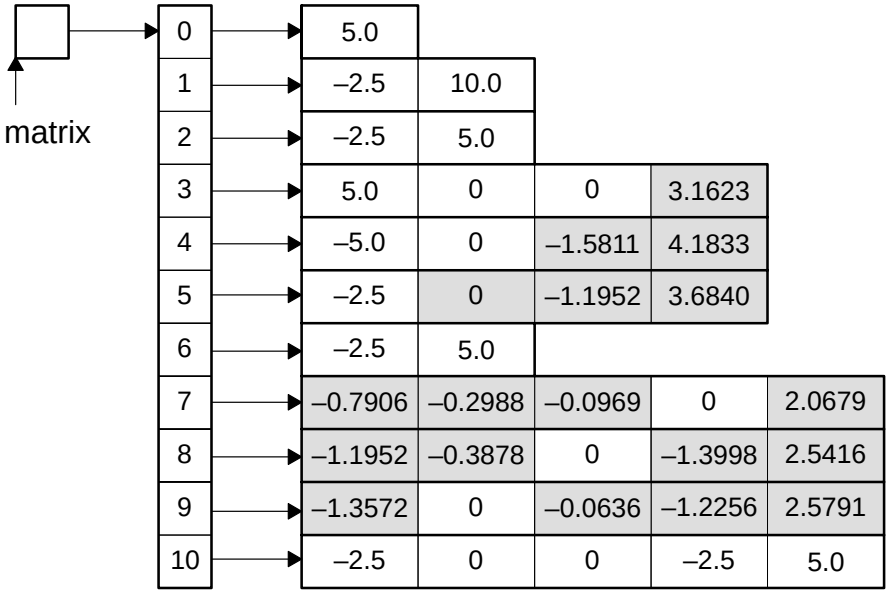
– In den Zeilen mit Status false wird $\mathbf{x}_1$ durch eine Vorwärts- und eine Rückwärtsauflösung bestimmt.

– In den Zeilen mit Status true wird $\mathbf{y}_2$ durch Substitution von $\mathbf{x}$ bestimmt.

Wegen der Datenstruktur der Systemmatrix wird die Substitution oberhalb und unterhalb der Diagonalen getrennt programmiert.

6.4.5 Rechenbeispiel

In dem Beispiel von Abschnitt 6.1.3 besteht die Matrix $\mathbf{A}_{11}$ aus den Zeilen und Spalten 3 bis 5 und 7 bis 9. Die Linksdreiecksmatrix der Zerlegung von $\mathbf{A}_{11}$ wird in die Datenstruktur der Systemmatrix eingetragen :



In dem folgenden Diagramm sind nur die Zeilen und Spalten der Matrix $\mathbf{A}_{11}$ und ihre Zerlegung gezeigt. Die bereits in der Datenstruktur der Systemmatrix gezeigten Koeffizienten der Zerlegung sind jetzt an ihren echten Positionen gezeigt.

	3	4	5	7	8	9
3	3.1623	–1.5811	0	–0.7906		
4		4.1833	–1.1952	–0.2988	–1.1952	
5			3.6840	–0.0969	–0.3878	–1.3572
7				2.0679	–1.3998	–0.0636
8					2.5416	–1.2256
9						2.5791

	3	4	5	7	8	9
3	3.1623					
4	–1.5811	4.1833				
5	0	–1.1952	3.6840			
7	–0.7906	–0.2988	–0.0969	2.0679		
8		–1.1952	–0.3878	–1.3998	2.5416	
9			–1.3572	–0.0636	–1.2256	2.5791

	10.0	–5.0		–2.5		
	–5.0	20.0	–5.0		–5.0	
		–5.0	15.0			–5.0
	–2.5			5.0	–2.5	
		–5.0		–2.5	10.0	–2.5
			–5.0		–2.5	10.0

Für den primalen Lösungsvektor werden folgende Bezeichnungen definiert :

$\mathbf{u}_{s1}$ Wert nach der Substitution der eingeprägten Variablen

$\mathbf{u}_{s2}$ Wert nach der Vorwärts – Auflösung

$\mathbf{u}_{s3}$ Wert nach der Rückwärtsauflösung

Im folgenden sind die Koeffizienten des primalen und des dualen Vektors gezeigt :

$\mathbf{u}_{s1} =$

10.0
10.0
10.0
25.0
60.0
80.0
20.0
0
0
50.0
20.0

$\mathbf{u}_{s2} =$

10.0000
10.0000
10.0000
7.9057
17.3308
27.3387
20.0000
6.8082
16.0688
41.5773
20.0000

$\mathbf{u}_{s3} =$

10.0000
10.0000
10.0000
12.5638
13.4631
15.1946
20.0000
13.3289
14.0940
16.0688
20.0000

$\mathbf{q}_{s} =$

6.4094
22.3154
22.9866
0
0
0
–12.0134
0
0
0
– 9.6980

In den folgenden Diagrammen ist die Berechnung von $\mathbf{u}_{s1}$ und $\mathbf{q}_{s}$ sowie die Kontrolle von $\mathbf{u}_{s3}$ gezeigt. Das negative Vorzeichen von $\mathbf{q}_{s}$ in (64) ist zu beachten.

			0	1	2	6	10				
3	0		–2.5	0					10.0		25.0
4	10.0			–5.0	0	–5.0			10.0		60.0
5	5.0				–2.5	–2.5		*	10.0	=	80.0
7	0	–							20.0		0
8	0								20.0		0
9	0										50.0
	$\{ \mathbf{w_s} \}$		$\{ \mathbf{A} \}$						$\{ \mathbf{u_s} \}$		$\{ \mathbf{u_{s1}} \}$

	3	4	5	7	8	9			
3	10.0	–5.0		–2.5				12.5638	25.0
4	–5.0	20.0	–5.0		–5.0			13.4631	60.0
5		–5.0	15.0			–5.0	*	15.1946	80.0
7	–2.5			5.0	–2.5			13.3289	0
8		–5.0		–2.5	10.0	–2.5	=	14.0940	0
9			–5.0		–2.5	10.0		16.1208	50.0
	$\{ \mathbf{A} \}$							$\{ \mathbf{u_{s3}} \}$	$\{ \mathbf{u_{s1}} \}$

	0	1	2	3	4	5	6	7	8	9	10				
0	5.0	–2.5		–2.5								10.0000		0	–6.4094
1	–2.5	10.0	–2.5	0	–5.0							10.0000		5.0	–22.3154
2		–2.5	5.0			–2.5						10.0000	–	10.0	= –22.9866
6						–2.5	5.0			0	–2.5	12.5638		0	12.0134
10							–2.5			–2.5	5.0	13.4631		0	9.6980
												15.1946			
												20.0000			
												13.3289			
												14.0940			
												16.1208			
												20.0000			
	$\{ \mathbf{A} \}$													$\{ \mathbf{W_s} \}$	$\{ \mathbf{u_{s3}} \}$

6.4.6 Klasse Model

Inhalt : Jedes Objekt dieser Klasse beschreibt ein Finite Element Modell für stationären Wärmefluß. Ein Modell besteht hauptsächlich aus HashSets, welche die Referenzen der Modellkomponenten enthalten, und aus Methoden zur Bearbeitung dieser Hash-Sets. Mit der Methode output( ) wird die aplhanumerische Ausgabe gesteuert.

Bestandteile : Ein Modell enthält einen HashSet für jede der in Abschnitt 6.3 beschriebenen Klassen von Modellkomponeten. Die HashSets werden im Konstruktor der Klasse Model definiert.

HashSet nodeSet
HashSet elementSet
HashSet materialSet
HashSet supportSet
HashSet nodeLoadSet
HashSet lineLoadSet
HashSet areaLoadSet

Methoden : Mit der Methode **add**(object) wird eine beliebige Komponente in das Modell eingebaut. Der Datentyp des Argumentes bestimmt den HashSet, in dem die Referenz des Objekttes eingetragen wird. Mit der Methode **remove**(identifier) wird eine beliebige Komponente im Modell gelöscht. Der Datentyp des Objektes, das mit diesem Identifikator in das Objektverzeichnis der Klase App eingetragen ist, bestimmt den HashSet, in dem die Referenz des Objektes gelöscht wird.

```
//....Add objects to Model.........................................
  public void add(Node      node    ) { nodeSet.add(node       );  }
  public void add(Element   element ) {elementSet.add(element  );  }
  public void add(Material  material) {materialSet.add(material);  }
  public void add(Support   support ) {supportSet.add(support  );  }
  public void add(NodeLoad  nodeLoad) {nodeLoadSet.add(nodeLoad);  }
  public void add(LineLoad  lineLoad) {lineLoadSet.add(lineLoad);  }
  public void add(AreaLoad  areaLoad) {areaLoadSet.add(areaLoad);  }
//....Remove objects from model....................................
  public boolean remove(String identifier)
  { String className                                    ;
    AppObject object = (AppObject)App.getObject(identifier) ;
    className        = object.getClass().getName()          ;
    if      (className.equals("Node"))
      return nodeSet.remove((Node)object)                 ;
    else if (className.equals("Element"))
      return elementSet.remove((Element)object)           ;
    else if (className.equals("Material"))
      return materialSet.remove((Material)object)         ;
    else if (className.equals("Support"))
```

```
      return supportSet.remove((Support)object)            ;
    else if (className.equals("NodeLoad"))
      return nodeLoadSet.remove((NodeLoad)object)          ;
    else if (className.equals("LineLoad"))
      return lineLoadSet.remove((LineLoad)object)          ;
    else if (className.equals("Areaload"))
      return areaLoadSet.remove((AreaLoad)object)          ;
    else return false                                      ;
  }
```

6.4.7 Klasse Equation

Datenstruktur : Jedes Gleichungssystem mit Profilstruktur kann unabhängig von seiner physikalischen Bedeutung als Objekt der Klasse Equation konstruiert werden. Die Dimension des Gleichungssystems wird bei der Konstruktion des Objektes festgelegt. Das Objekt enthält folgende Datenfelder für das Gleichungssystem :

int	profil []	Profilvektor
double	matrix [][]	Systemmatrix $\mathbf{K}_S$
double	vector []	Systemvektor $\mathbf{w}_S$
double	primal []	Primaler Variablenvektor $\mathbf{u}_S$
double	dual []	Dualer Variablenvektor $\mathbf{q}_S$
boolean	status []	Einprägung : true := primale Variable

Methoden : Das Gleichtungssystem wird mit folgenden Methoden aufgestellt und gelöst :

Equation (int dimension)	
void	setProfile (int index [])
void	setStatus (boolean status, int index, double value)
void	allocateMatrix ( )
void	addMatrix (int index [], double submatrix [][])
void	addVector (int index [], double subvektor [])
void	addScalar (int index, double value)
void	decompose ( )
void	solve ( )

Diese Methoden und ihre Beziehungen zu den Datenstrukturen sind in den folgenden Abschnitten behandelt.

```
//-------------------------------------------------------------
//  CLASS : Equation                Linear System of Equations
//-------------------------------------------------------------
//  FUNCTION :
//
//  Construction and solution of a linear system of
//  equations with profile structure :
//
//      A * u = w + q
//
//  A   system matrix with given coefficients
/// u   primale soluntion vector
//  q   dual solution vector
//  w   system vector with given coefficients
//
//  In each row, either u[i] or q[i] is given .
//
//-------------------------------------------------------------
//  METHODS :
//
//        Equation  (int dimension)       Constructor
//  void  setProfile(int index[])         set for one element
//  void  setStatus()                     set for one node
//  void  allocateMatrix()                rows of system matrix
//  void  decompose()                     decompose A
//  void  solve()                         solve the equations
//  void  addMatrix(int index[],double matrix[][])
//  void  addVector(int index[],double vectro[]  )
//
//-------------------------------------------------------------

class Equation
{ static  final double EPSILON = 1.0e-20 ;

  boolean status[]   ;           // true  : primal prescribed

// false : dual   prescribed
  int     profile[]  ;           // index of 1. column != 0
  double  matrix[][] ;           // system matrix A
  double  vector[]   ;           // system vektor w
  double  primal[]   ;           // primal solution vector
  double  dual[]     ;           // dual   solution vector

  int     row, column, dimension ;

//..Construction of the system of equations...................

  Equation(int n)
  { dimension = n                        ;
    status    = new boolean[dimension]   ;
    profile   = new int    [dimension]   ;
    primal    = new double [dimension]   ;
    dual      = new double [dimension]   ;
    vector    = new double [dimension]   ;
    matrix    = new double [dimension][] ;
    for (row = 0; row < dimension; row++)
    { profile[row] = row;
  } }

//..Set the profile vector for one element....................
```

```
  void setProfile(int index[])
  { for (int k=0; k < index.length; k++)
    { for (int m=0; m < index.length; m++)
      { if (profile[index[k]] > index[m])
            profile[index[k]] = index[m] ;
        if (profile[index[m]] > index[k])
            profile[index[m]] = index[k] ;
  } } }
//..Set the status vector for one node.....................
  void setStatus(boolean status, int index, double value)
  { if (status)
    { this.status[index] = true  ;
      this.primal[index] = value ;
      return ;
    }
    dual[index] = value ;
  }
//..Allocate the rowvectors of the system matrix.............
  void allocateMatrix()
  { for (row = 0; row < dimension; row++)
    { matrix[row] = new double[row – profile[row] + 1] ;
  } }
//..Read/write a coefficient in the system matrix.............
  double getValue(int i,int m)
  { return matrix[i][m – profile[i]]; }
  void setValue(int i,int m, double value)
  { matrix[i][m – profile[i]] = value; }
  void addValue(int i,int m, double value)
  { matrix[i][m – profile[i]] += value; }
//..Add matrix to system matrix..............................
  void addMatrix(int index[], double matrix[][])
  { for (int k=0; k < index.length; k++)
    { for (int m=0; m < index.length; m++)
      { if (index[m] > index[k]) continue;
        addValue(index[k], index[m], matrix[k][m]) ;
  } } }
//..Add vector to system vector..............................
  void addVector(int index[], double vector[])
  { for (int k=0; k < index.length; k++)
    { this.vector[index[k]] += vector[k] ;
  } }
//..Add scalar to system vector..............................
  void addScalar(int index, double value)
  { vector[index] += value ;
  }
//__DECOMPOSE THE SYSTEM MATRIX_____________________________
  void decompose() throws AlgebraicException
  { int    start ;
    double sum   ;
//..A[i][m] = A[i][m] – Sum(A[i][k]*A[k][m]) / A[k][k].........
```

```
    for (row = 0; row < dimension; row++)
    { if (status[row]) continue ;
      for (column = profile[row]; column < row; column++)
      { if (status[column]) continue ;
        start = Math.max(profile[row],profile[column]);
        sum   = getValue(row,column) ;
        for (int m = start; m < column; m++)
        { if (status[m]) continue ;
          sum –= getValue(row,m) * getValue(column,m) ;
        }
        sum /= getValue(column,column) ;
        setValue(row,column,sum) ;
      }
//..A[i][i] = sqr{(A[i][i] – Sum(A[i][m]*A[m][i])}.............

      sum = getValue(row,row) ;
      for (int m = profile[row]; m < row; m++)
      { if (status[m]) continue ;
        sum –= getValue(row,m) * getValue(row,m) ;
      }
      if (sum < EPSILON) throw new AlgebraicException
          (" Element <= 0 in decomposition of row " + row) ;
      setValue(row,row,Math.sqrt(sum)) ;
  } }

//__SOLVE THE SYSTEM EQUATIONS_____________________________________

//..Substitute the prescribed variables in the rows without
//  prescribed primary variables : u = c1 + y1 – A12 * x2

  void solve()
  { for (row = 0; row < dimension; row++)
    { if (status[row]) continue;
      primal[row] = vector[row] + dual[row] ;
      for (column = profile[row]; column < row; column++)
      { if (! status[column]) continue;
      primal[row] –= getValue(row,column) * primal[column];
    } }

    for (column = 0; column < dimension; column++)
    { if (! status[column]) continue;
      for (row = profile[column]; row < column; row++)
      { if (status[row]) continue;
        primal[row] –= getValue(column,row) * primal[column];
    } }

//..Compute primal variables : forward sweep by rows..........

    for (row = 0; row < dimension; row++)
    { if (status[row]) continue;
      for (column = profile[row]; column < row; column++)
      { if (status[column]) continue;
        primal[row] –= getValue(row,column) * primal[column];
      }
      primal[row] /= getValue(row,row) ;
    }

//..Compute primal variables : backward sweep by rows.........

    for (column = dimension–1; column >= 0; column––)
    { if (status[column]) continue ;
      primal[column] /= getValue(column,column) ;
```

```
      for (row = profile[column]; row < column; row++)
      { if (status[row]) continue ;
        primal[row] –= getValue(column,row) * primal[column];
    } }
//..Compute dual variables : substitute the primal variables...
//  in the rows with prescribed primal variables :
    for (row = 0; row < dimension; row++)
    { if (! status[row]) continue;
      dual[row] = –vector[row] ;
      for (column = profile[row]; column <= row; column++)
      { dual[row] += getValue(row,column) * primal[column];
    } }
    for (column = 0; column < dimension; column++)
    { for (row = profile[column]; row < column; row++)
      { if (! status[row]) continue ;
        dual[row] += getValue(column,row) * primal[column] ;
} } } }
//.....................................................
```

6.4.8 Klasse Analysis

Inhalt : Jedes Objekt dieser Klasse wird zur Berechnung der stationären Wärmeströmung in einem Modell eingesetzt. Die zentrale Methode perform() der Klasse steuert die Berechnungsschritte :

```
void perform() throws FemException, AlgebraicException
{ setReferences()        ;
  computeSystemMatrix()  ;
  computeSystemVector()  ;
  setStatusVector()      ;
  system.decompose()     ;
  system.solve()         ;
  saveResult()           ;
  isPerformed = true     ;
}
```

Referenzen : Die Methode setReferences() traversiert alle HashSets des Modells. Für jedes Objekt dieser Mengen wird die Methode setReferences() aufgerufen. Diese Methode bestimmt die Referenzen aller Objekte, deren Identifikatioren im Objekt enthalten sind, und speichert diese Referenzen im Objekt. Fehlende Objekte werden gemeldet. Nach Ausführung der Methode setReferences() der Klasse Analysis sind alle Objekte des Modells verknüpft.

Aufbau der Systemmatrix : Die Systemmatrix $\mathbf{K}_S$ wird in der Methode computeSystemMatrix( ) aufgebaut. Die Datenstruktur von $\mathbf{K}_S$ ist zunächst unbekannt. Daher gliedert sich der Aufbau von $\mathbf{K}_S$ in folgende Schritte :

(1) Bestimme die Dimension der Systemgleichungen und konstruiere ein Objekt der Klasse Equation mit dieser Dimension.

(2) Bestimme den Profilvektor der Systemmatrix (siehe Abschnitt 6.4.2) und konstruiere die Zeilenvektoren von $\mathbf{K}_S$ durch Aufruf der Methode allocateMatrix( ) der Klasse Equation.

(3) Bilde eine Schleife über alle Element-Objekte. Bestimme für jedes Element die Elementmatrix durch Aufruf der Methode addMatrix( ) der Klasse Equation.

```
void computeSystemMatrix() throws FemException
{ dimension = 0                                ;

  // determine the dimension of the system equations
  iter = model.nodeSet.iterator()              ;
  while (iter.hasNext())
  { node = (Node)iter.next()                   ;
    if (node.index > dimension)
        dimension = node.index                 ;
  }
  dimension++                                  ;
  system = new Equation(dimension)             ;

  // set the profile of the system equations
  iter = model.elementSet.iterator()           ;
  while (iter.hasNext())
  { element = (Element)iter.next()             ;
    indexList = element.getSystemIndices()     ;
    system.setProfile(indexList)               ;
  }
  system.allocateMatrix()                      ;

  // set the coefficients of the system equations
  iter = model.elementSet.iterator()           ;
  while (iter.hasNext())
  { element = (Element)iter.next()             ;
    indexList = element.getSystemIndices()     ;
    elementMatrix = element.computeMatrix()    ;
    system.addMatrix(indexList,elementMatrix) ;
} }
```

Aufbau des Systemvektors : Der Systemvektor $\mathbf{w}_S$ in den Systemgleichungen (64) wird mit der Methode computeSystemVector( ) bestimmt. Die Beiträge der Knotenlasten, Linienlasten und Flächenlasten werden in getrennten Traversen durch die Hash-Sets dieser Komponenten bestimmt. Bei der Addition des Elementbeitrages $\mathbf{R}_e^T\mathbf{w}_e$ wird die Multiplikation mit der Topologiematrix $\mathbf{R}_e^r$ analog zum Aufbau der Systemmatrix durch die Abbildung der Elementindizes auf die Systemindizes ersetzt.

```
void computeSystemVector() throws FemException
{ Node  nodeList[]                              ;
  double  value, vector[]                       ;
  int     indexList[]                           ;
  iter = model.nodeLoadSet.iterator()           ; // node loads
  while (iter.hasNext())
  { nodeLoad = (NodeLoad)iter.next()            ;
    node     =  nodeLoad.getNodeReference()     ;
    index    =  node.getSystemIndex()           ;
    value    =  nodeLoad.getStream()            ;
    system.addScalar(index,value)               ;
  }
  iter = model.lineLoadSet.iterator()           ; // line loads
  while (iter.hasNext())
  { lineLoad = (LineLoad)iter.next()            ;
    vector   = lineLoad.computeLoadVector()     ;
    node     =  lineLoad.node[0]                ;
    index    =  node.getSystemIndex()           ;
    system.addScalar(index,vector[0])           ;
    node     =  lineLoad.node[1]                ;
    index    =  node.getSystemIndex()           ;
    system.addScalar(index,vector[1])           ;
  }
  iter = model.areaLoadSet.iterator()           ; // area loads
  while (iter.hasNext())
  { areaLoad   = (AreaLoad)iter.next()          ;
    vector     = areaLoad.computeLoadVector()   ;
    indexList  = areaLoad.getSystemIndices()    ;
    system.addVector(indexList,vector)          ;
} }
```