FEJavaDemo

1. Betrachtete Aufgabeklasse

Mittels der Methode der finiten Elemente kann FEJavaDemo Wärmeleitprobleme lösen.
Gesucht ist das stationäre Temperaturfeld *u(x)* in einem zweidimensionalen Gebiet *Ω*, das aus mehreren Materialbereichen mit verschiedenen Wärmeleitzahlen *Λ(x)* bestehen kann. Im Gebiet kann es eine Wärmequelle geben, die durch eine Funktion *f* beschrieben wird.

Drei Arten von Randbedingungen können betrachtet werden:

- Die Dirichletschen Randbedingungen (Randstück Γ_1): Teil des Gebietsrandes, auf dem Werte $g_1(x)$ für die Temperatur vorgegeben sind.
- Die Neumannschen Randbedingungen (Randstück Γ_2): Teil des Gebietsrandes, auf dem der Wärmefluß $g_2(x)$ vorgegeben ist (Mit $g_2(x)=0$ kann auf Randstücken eine Wärmeisolation modelliert werden)
- Die Robinschen Randbedingungen (Randstück Γ_3): Modellierung des freien Wärmeaustauschs mit der Umgebung (der Wärmefluß ist proportional ($\alpha(x)$ Koeffizient) zur Differenz zwischen dem Wert der Temperatur am Rand $u(x)$ und der Umgebungstemperatur $u_A(x)$)

Klassische Formulierung:

Gesucht ist $u \in C^2(\Omega) \cap C^1(\Omega \cup \Gamma_2 \cup \Gamma_3) \cap C(\bar{\Omega})$, so dass

$$
\begin{aligned}
-\text{div}(\Lambda(x)\ \text{grad}\ u(x)) &= f(x) && \forall x \in \Omega\ , \\
u(x) &= g_1(x) && \forall x \in \Gamma_1\ , \\
\frac{\partial u}{\partial N} &= g_2(x) && \forall x \in \Gamma_2\ , \\
\frac{\partial u}{\partial N} &= \alpha(x)\ \left[u_A(x) - u(x)\right] && \forall x \in \Gamma_3\ ,
\end{aligned}
$$

gilt mit $\Gamma = \partial\Omega = \bar{\Gamma}_1 \cup \bar{\Gamma}_2 \cup \bar{\Gamma}_3$ und $\Gamma_i \cap \Gamma_j = \emptyset$ für $i \neq j$

und $\Lambda(x) = \begin{pmatrix} \lambda_1(x) & 0 \\ 0 & \lambda_2(x) \end{pmatrix}$ mit $\frac{\partial u}{\partial N} = \lambda_1(x)\frac{\partial u}{\partial x_1} n_1 + \lambda_2(x)\frac{\partial u}{\partial x_2} n_2$

und $\vec{n}(x) = \begin{pmatrix} n_1(x) \\ n_2(x) \end{pmatrix}$ der Vektor der äußeren Einheitsnormalen im Punkt x ∈ Γ.

Im Programm wird folgendes an die Eingangsdaten vorausgesetzt:

- Die Funktion kann auf verschiedenen Bereichen verschiedene Definitionen haben. Dazu kann sie mittels Polynome, tan(), sin(), cos(), exp(), ln(), sqrt() und der Konstante π definiert werden, zum Beispiel f(x,y) = 2*cos(Pi*x*y).
- Λ wird mittels λ_1 und λ_2 definiert. Diese Koeffizienten sind Konstanten, die verschieden in jedem Materialbereich sein können.
- Was die Dirichletschen Randbedingungen betrifft, wird die Temperatur $g_1(x)$ nur auf einigen Punkten vorgegeben. Diese Punkte sind die so genannten Dirichlet-Knoten.
- Die anderen Funktionen $g_2(x)$, $\alpha(x)$ und $u_A(x)$ in den Randbedingungen 2. und 3. Art sind auf jeder Kante konstant. So kann man auch sagen, dass sie stückweise konstant sind.

Variationsformulierung der Aufgabe:

Gesucht ist $u \in V_{g_1}$, so dass

$$a(u,v) = \langle F, v \rangle \qquad \forall v \in V_0$$

gilt mit

$$a(u,v) = \int_\Omega (\text{grad } v(x))^T \Lambda(x) \text{ grad } v(x)\, dx + \int_{\Gamma_3} \alpha(x) u(x) v(x) ds ,$$

$$\langle F, v \rangle = \int_\Omega f(x) v(x) dx + \int_{\Gamma_2} g_2(x) v(x) ds + \int_{\Gamma_3} \alpha(x) u_A(x) v(x) ds ,$$

$$V_{g_1} = \left\{ u \in H^1(\Omega) : u(x) = g_1(x) \text{ auf } \Gamma_1 \right\},$$

$$V_0 = \left\{ v \in H^1(\Omega) : v(x) = 0 \text{ auf } \Gamma_1 \right\}.$$

Galerkin-Verfahren

Die Grundidee des Galerkin-Verfahrens besteht darin, den unendlichdimensionalen Raum der Grundfunktionen durch einen endlichdimensionalen Raum V_h zu ersetzen:

$$V_h = \left\{ v_h \ : \ v_h = \sum_{i \in \bar{\omega}_h} v_i p_i(x) \right\} = span\{ p_i : i \in \bar{\omega}_h \} \ \subset \ V = H^1(\Omega)$$

Dabei bezeichnet $\bar{\omega}_h$ die Indexmenge, welche die Nummern aller Knoten enthält.

Die vorgegebenen Funktionen p_i sind die so genannten linear unabhängigen Ansatzfunktionen, linear unabhängig, und die Koeffizienten v_i sind frei wählbar.

Da wir die Werte der Lösung in den Dirichlet-Knoten kennen, suchen wir eine Näherungslösung der Aufgabe in der Form

$$u_h(x) = \sum_{j \in \omega_h} u_j p_j(x) + \sum_{j \in \gamma_h} u_{*,j} p_j(x)$$

wo nur die u_j unbekannt sind.

Hierbei ist ω_h die Indexmenge, welche die Nummern der Knoten in $\Omega \cup \Gamma_2 \cup \Gamma_3$ enthält und γ_h die Indexmenge, welche die Nummern der Knoten auf $\bar{\Gamma}_1$ enthält: $\bar{\omega}_h = \omega_h \cup \gamma_h$.

Damit ergibt sich das Gleichungssystem:

Gesucht ist $\underline{u}_h = [u_j]_{j \in \omega_h} \in \mathbb{R}^{N_h}$:

$$\sum_{j \in \omega_h} u_j a(p_j, p_i) = \langle F, p_i \rangle - \sum_{j \in \gamma_h} u_{*,j} a(p_j, p_i) \qquad \forall i \in \omega_h$$

d.h. gesucht ist $\underline{u}_h = [u_j]_{j \in \omega_h} \in \mathbb{R}^{N_h}$:

$$K_h \underline{u}_h = \underline{f}_h$$

mit

$$K_h = [a(p_j, p_i)]_{i,j \in \omega_h}$$

$$\underline{f}_h = \left[\langle F, p_i \rangle - \sum_{j \in \gamma_h} u_{*,j} a(p_j, p_i) \right]_{i \in \omega_h} \in \mathbb{R}^{N_h}$$

N_h ist die Anzahl der Knoten, die in $\Omega \cup \Gamma_2 \cup \Gamma_3$ liegen.

2. FE-Diskretisierung

2.1. Art der Vernetzung und Netzverfeinerung

Wir zerlegen das Gebiet Ω, in dem die Lösung des betrachteten Randwertproblems gesucht wird, in finite Elemente $T^{(r)}$. In FEJavaDemo können dies Dreiecke oder Vierecke sein.

Falls das Programm mit Dreiecken und stückweise linearen Ansatzfunktionen p_i arbeitet, kann auch die Vernetzung verfeinert werden. Nach der Verfeinerung wird eine erneute Problembearbeitung mit der feineren Vernetzung durchgeführt. Wenn eine Verfeinerung gefordert wird, wird eine höchste zugelassene Temperaturabweichung abgefragt. Das Programm verfeinert die Vernetzung, bis alle Temperaturabweichungen zwischen zwei Eckpunkten jedes Dreiecks unter einem gewünschten Grenzwert liegen. Falls „0“ eingegeben wurde, wird die Vernetzung vollständig verfeinert (jedes Dreieck wird in vier kongruente Teildreiecke geteilt) aber nur ein Mal.

Algorithmus der Netzverfeinerung:

(1) Teile alle Dreiecke, für die die Temperaturabweichung zwischen 2 Knoten zu hoch ist, in vier kongruente Teildreiecke, d.h. definiere auf allen Kanten dieser Dreiecke den Kantenmittelpunkt und generiere durch Verbindung dieser Mittelpunkte die Teildreiecke.

(2) Solange mindestens eine Dreiecksviertelungen durchgeführt wurde

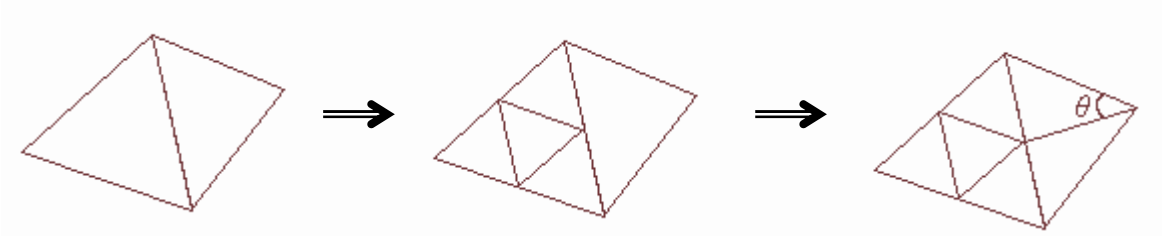
   (i) Teile im zu verfeinernden Netz alle Dreiecke, die nicht geviertelt wurden, bei denen aber auf mindestens zwei Kanten neue Knoten generiert worden sind, in vier kongruente Teildreiecke.

   (ii) Für jedes Dreieck, bei dem genau auf einer Kante ein neuer Knoten generiert wurde,

- Wenn bei einer Bisektion dieses Dreiecks 2 Dreiecke, die eine schlechte Qualität hätten (d.h. die Innenwinkel θ wäre zu spitz: $\theta<25°$), generiert würden, führe eine Dreiecksviertelung dieses Element durch.

- Sonst teile dieses Dreieck in zwei Teildreiecke, indem dieser Knoten mit dem der entsprechenden Kante gegenüberliegenden Knoten verbunden wird.

(3) Löse das Problem unter Nutzung der neuen Vernetzung. Wenn es noch Temperaturabweichungen über dem gewünschte Grenzwert gibt, gehe zu Schritt (1) zurück



Netzverfeinerung mit Dreiecksviertelung

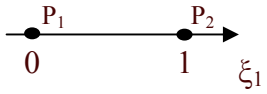
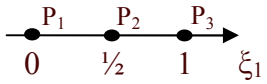
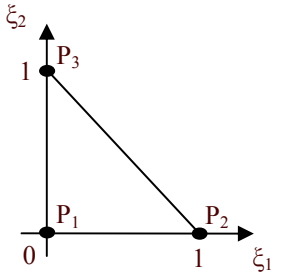
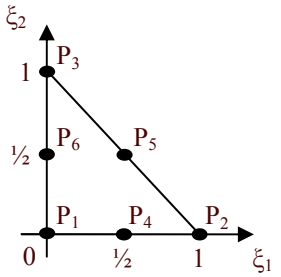
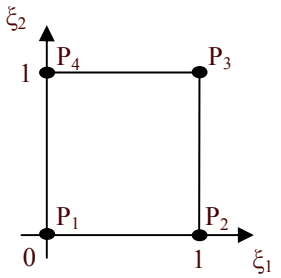
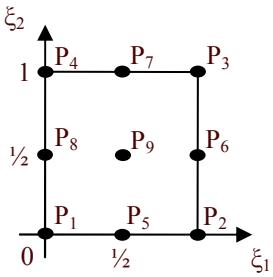
2.2. Wahl der Ansatzfunktionen

Als Formfunktionen über dem Referenzelement wählen wir polynomiale Funktionen φ_α, für die in den Knoten P_β des Referenzelementes die Bedingung $\varphi_\alpha(\xi_\beta) = \delta_{\alpha\beta}$ erfüllt ist.

Das Programm kann mit linearen und quadratischen Funktionen arbeiten.

Für die Berechnung der Einträge der Steifigkeitsmatrix und des Lastvektors sind nur die auf dem Referenzelement definierten Formfunktionen nötig, weil die entsprechenden Integrale auf Integrale über dem Referenzelement zurückgeführt werden. Dafür wird eine Transformation, die die Abbildung des Referenzelements auf ein beliebiges Element $T^{(r)}$ der Vernetzung realisiert, genutzt.

Die folgenden Formfunktionen werden genutzt:

	$\varphi_1^l(\xi) = -\xi_1 + 1$ $\varphi_2^l(\xi) = \xi_1$
	$\varphi_1^q(\xi) = 2\xi^2 - 3\xi + 1$ $\varphi_2^q(\xi) = -4\xi^2 + 4\xi$ $\varphi_3^q(\xi) = 2\xi^2 - \xi$
	$\varphi_1^{l2}(\xi_1,\xi_2) = 1-\xi_1-\xi_2$ $\varphi_2^{l2}(\xi_1,\xi_2) = \xi_1$ $\varphi_3^{l2}(\xi_1,\xi_2) = \xi_2$
	$\varphi_\alpha^{q2}(\xi_1,\xi_2) = \varphi_\alpha^{l2}\left(2\varphi_\alpha^{l2}-1\right), \quad \alpha = 1,2,3$ $\varphi_4^{q2}(\xi_1,\xi_2) = 4\varphi_1^{l2}\varphi_2^{l2}$ $\varphi_5^{q2}(\xi_1,\xi_2) = 4\varphi_2^{l2}\varphi_3^{l2}$ $\varphi_6^{q2}(\xi_1,\xi_2) = 4\varphi_3^{l2}\varphi_1^{l2}$
	$\varphi_1^{bl}(\xi_1,\xi_2) = \varphi_1^l(\xi_1)\varphi_1^l(\xi_2)$ $\varphi_2^{bl}(\xi_1,\xi_2) = \varphi_2^l(\xi_1)\varphi_1^l(\xi_2)$ $\varphi_3^{bl}(\xi_1,\xi_2) = \varphi_2^l(\xi_1)\varphi_2^l(\xi_2)$ $\varphi_4^{bl}(\xi_1,\xi_2) = \varphi_1^l(\xi_1)\varphi_2^l(\xi_2)$
	$\varphi_1^{bq}(\xi_1,\xi_2) = \varphi_1^q(\xi_1)\varphi_1^q(\xi_2)$ $\varphi_2^{bq}(\xi_1,\xi_2) = \varphi_3^q(\xi_1)\varphi_1^q(\xi_2)$ $\varphi_3^{bq}(\xi_1,\xi_2) = \varphi_3^q(\xi_1)\varphi_3^q(\xi_2)$ $\varphi_4^{bq}(\xi_1,\xi_2) = \varphi_1^q(\xi_1)\varphi_3^q(\xi_2)$ $\varphi_5^{bq}(\xi_1,\xi_2) = \varphi_2^q(\xi_1)\varphi_1^q(\xi_2)$ $\varphi_6^{bq}(\xi_1,\xi_2) = \varphi_3^q(\xi_1)\varphi_2^q(\xi_2)$ $\varphi_7^{bq}(\xi_1,\xi_2) = \varphi_2^q(\xi_1)\varphi_3^q(\xi_2)$ $\varphi_8^{bq}(\xi_1,\xi_2) = \varphi_1^q(\xi_1)\varphi_2^q(\xi_2)$ $\varphi_9^{bq}(\xi_1,\xi_2) = \varphi_2^q(\xi_1)\varphi_2^q(\xi_2)$

2.3. Speicherungstechnik und Umnummerierung der Knoten

Aufgrund der Wahl der Ansatzfunktionen mit lokalem Träger ist die Steifigkeitsmatrix schwach besetzt. Die Matrixeinträge $K_{ij} = a(p_j,p_i)$ sind nämlich nur von Null verschieden, wenn der Durchschnitt des Inneren der Träger der Ansatzfunktionen p_i und p_j nicht leer ist.

Bei geeigneter Knotennummerierung besitzen die FE-Matrizen eine Bandstruktur. Die Bandweite, d.h. der größte Abstand, den ein Nicht-Null-Element einer Spalte vom entsprechenden Hauptdiagonalelement hat, hängt von der Knotennummerierung ab.

Diese zwei Eigenschaften werden bei der Abspeicherung von K genutzt. Das Programm benutzt die „*variable Bandweite spaltenweise (VBS)*“ oder „*Skyline-Speicherung*“. Dabei werden nur die Elemente abgespeichert, die in der Hülle sind, d.h. zwischen dem ersten Nicht-Null-Element der jeweiligen Spalte und dem Hauptdiagonalelement.

Von der Nummerierung der Knoten hängt es ab, wie groß die Besetztheitsstruktur und damit die Zahl der Elemente, die abgespeichert werden, von der Knotennummerierung ab. Es ist wichtig, dass die Knoten jedes Elements Nummern haben, so dass die Differenz der Knotennummern relativ klein ist. Dies führt zu einer „kleinen“ Länge der Hülle.

Für eine gegebene Knotennummerierung kann das Programm (mit dem Algorithmus von Cuthill-McKee) eine Umnummerierungsstrategie durchführen, um die Länge der Hülle zu reduzieren.

Das Programm kann .net Dateien, die für lineare Ansatzfunktionen geschrieben sind, auch bei quadratischen (oder kubischen, ...) Funktionen benutzen. In diesem Fall sind nur die 3 Eckpunkte nummeriert und das Programm fügt die zusätzlichen Knoten (3 pro Dreieck für die quadratischen Funktionen) mit den folgenden Nummern hinzu. Es ist aber möglich, die Knoten automatisch umzunummerieren, um den Prozess der Auflösung des Problems schneller zu machen.

Algorithmus von Cuthill McKee:

Die heuristische Begründung des beschriebenen Vorgehens besteht einfach darin, dass benachbarte Knoten möglichst bald im Nummerierungsprozess berücksichtigt werden müssen, andernfalls große Indexdifferenzen auftreten, was zu einer großen Bandbreite führt.
Deshalb beruht beispielsweise die Festsetzung, die Knoten innerhalb einer Stufe fortlaufend unter Berücksichtigung ihres zunehmenden Grades (Unter dem Grad $d(z)$ eines Knotens z versteht man

die Anzahl der benachbarten Knoten) zu nummerieren, auf der einleuchtenden Strategie, dass Knoten mit vielen Nachbarn möglichst hohe Nummern erhalten sollen, um die im nächstfolgenden Schritt auftretenden Indexdifferenzen klein zu halten.

1. Zum Startknoten *r* bestimmt man alle benachbarten Knoten. Diese Knoten werden mit zunehmendem Grad fortlaufend nummeriert. Bei benachbarten Knoten mit gleichem Grad besteht selbstverständlich eine Willkür in der Reihenfolge der Nummerierung. Die in diesem Schritt nummerierten Knoten haben alle die Distanz 1 vom Startknoten *r*. Sie bilden die erste Stufe.

2. Zu den Knoten der ersten Stufe mit aufsteigenden (neuen) Nummern bestimme man sukzessive ihre benachbarten und noch nicht neu nummerierten Knoten und nummeriere sie je mit zunehmendem Grad. Die in diesem Schritt nummerierten Knoten besitzen die Distanz 2 vom Startknoten und bilden die zweite Stufe im Nummerierungsprozess.
   Wiederhole diesen Prozess, bis alle Knoten durchnummeriert sind.

3. Man stellt fest, dass das Profil oft ganz wesentlich verkleinert werden kann, falls die Knotenvariablen exakt in der umgekehrten Reihenfolge durchnummeriert werden, wie sie der oben beschriebene Prozess liefert.

Implementierung des Algorithmus:

(1) Suche den so genannten Startknoten oder Wurzel *r*. Er bekommt den Nummer 1.
   $S = \{r\}$

(2) Solange nicht alle Knoten neunummeriert sind,
   - $S_{neu} = \{\}$
   - Für jeden Knoten *x* von *S*,
     - Zum Knoten *x* bestimme man alle benachbarten und noch nicht neu nummerierten Knoten $\{k_1, k_2, ..., k_r\}$ und nummeriere sie je mit zunehmendem Grad.
     - $S_{neu} = S_{neu} + \{k_1, k_2, ..., k_r\}$
   - $S = S_{neu}$

(3) Umkehrung der Nummerierung vermittels der Substitution: $k \rightarrow n+1-k$

Algorithmus zur Bestimmung der Startknote:

Im diesem Schritt zerlegt man die Vernetzung in Stufen oder Schichten.
$R(g) = (L_1, L_2, ..., L_r)$ heißt Stufenstruktur oder Schichtung (engl. level structure) mit der Wurzel g, wenn das Folgende gilt:

1. $\bigcup_{i=1}^{r} L_i$ = Menge, welche alle Knoten der Vernetzung enthält
2. L_r ist nichtleer
3. $L_1 = \{g\}$
4. Der kürzer Weg zwischen einem Knoten $h \in L_i$ und g besteht aus i-1 Kanten

r heißt die Tiefe der Schichtung. Ihre Breite k ist die Anzahl der Elemente der Schicht mit den meisten Knoten.

Gibbs, Poole und Stockmeyer haben vorgeschlagen, Stufenstrukturen mit geringer Breite zu benutzen. Sie werden gewöhnlich unter den Stufenstrukturen mit größter Tiefe gesucht. Das heißt, wenn mehr Stufen vorhanden sind, entfallen im Schnitt weniger Knoten auf die einzelnen Stufen.
Es wäre ideal, zuerst einen so genannten Durchmesser des Netzes zu bestimmen, der durch zwei Knoten festgelegt wird, deren kürzester Verbindungsweg im Graphen am längsten ist. Das Problem ist zwar lösbar, meist wird jedoch ein approximativer Algorithmus dafür genutzt.
Mit diesem Algorithmus bildet g nicht mit Sicherheit der Endpunkt eines Durchmessers, doch wird die Strukturtiefe annähernd maximal sein, und man begnügt sich mit einem Pseudodurchmesser.
Dieser erste Schritt ist sehr geeignet, automatisch günstige Startknoten zu bestimmen, da der Aufwand relativ gering ist und in der Regel nur wenige Startpunkte getestet werden müssen, um einen Pseudodurchmesser zu finden.

(1) Wähle einen Knoten g mit minimalem Grad

(2) Erzeuge die Schichtung $R(g) = (L_1, L_2, ..., L_r)$. Die Elemente der letzten Schicht L_r seien f_j. Sie seien bez. ihres Grades geordnet, d.h. $d(f_j) \leq d(f_{j+1})$

(3) Setze $j = 1$

(4) Berechne die Tiefe der Schichtung $R(f_j)$: s. Wenn $s > r$ ist, setze man $g = f_j$ und kehre nach (2) zurück

(5) Wenn $j < l$ ist, erhöhe man j um 1 und wiederhole (4)

2.4. Berechnung des FE-Gleichungssystems

2.4.1. Berechnung und Assemblierung der Steifigkeitsmatrix und des Lastvektors

Der Aufbau des FE-Gleichungssytems wird elementweise durchgeführt. Zuerst assembliert das Programm die Steifigkeitsmatrix und den Lastvektor ohne Berücksichtigung der Randbedingungen.

Bei der Berechnung der Elementsteifigkeitsmatrizen $K^{(r)}$ führen wir in den Integralen eine Variablensubstitution durch, so dass Integrale über dem Referenzelement $\hat{T}$ zu berechnen sind.

Für das Elements $T^{(r)}$ erhalten wir die Elementsteifigkeitsmatrix $K^{(r)}$:

$$K^{(r)} = \left[\int_{\hat{T}} \left[\left(grad_{\xi} \varphi_i(\xi) \right)^T \left(J^{(r)} \right)^{-1} \Lambda \left(J^{(r)} \right)^{-T} grad_{\xi} \varphi_j(\xi) \right] \left| \det J^{(r)} \right| d\xi \right]_{i,j=1}^{\hat{N}}$$

mit:

- ξ : Koordinate über dem Referenzelement
- φ_i und φ_j sind die auf dem Referenzelement definierten Formfunktionen
- $\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$: Wärmeleitzahlen des Materials
- $\hat{N}$: Anzahl der Knoten pro Element
- $J^{(r)}$: Jacobi-Matrix der Transformation, die die Abbildung des Referenzelements $\hat{T}$ auf ein beliebiges Element $T^{(r)}$ der Vernetzung realisiert.

Nur die Matrizen $J^{(r)}$, $(J^{(r)})^{-1}$ und die Ableitungen der über dem Referenzelement definierten Formfunktionen werden bei der Berechnung der Einträge der Elementsteifigkeitsmatrizen benötigt. Explizite Formeln für die Formfunktionen über den Elementen $T^{(r)}$ sind nicht erforderlich

Die Elementlastvektoren $f^{(r)}$ werden auf analoge berechnet, d.h.

$$\underline{f}^{(r)} = \left[\int_{\hat{T}} f\left(x_{T^{(r)}}(\xi) \right) \varphi_i(\xi) \left| \det J^{(r)} \right| d\xi \right]_{i=1}^{\hat{N}}$$

mit: $x_{T^{(r)}}(\xi) = J^{(r)} \xi + x_1^{(r)}$: Koordinaten über dem Element $T^{(r)}$

Während dieses ersten Prozesses werden auch die Steifigkeitsmatrix und der Lastvektor modifiziert, um die Randbedingungen 1. Art zu berücksichtigen.

Die Steifigkeitsmatrix wird durch

$$K_{ij}^{(r)} = K_{ji}^{(r)} = \delta_{ij} \qquad \forall i \in \gamma_h^{(r)}, j \in \bar{\omega}_h^{(r)}$$

korrigiert, mit $\bar{\omega}_h^{(r)}$: Indexmenge, welche die Nummern der Knoten im Element $T^{(r)}$ enthält.

Wir korrigieren auch die rechte Seite durch:

$$f_i^{(r)} := f_i^{(r)} - \sum_{j \in \gamma_h^{(r)}} K_{ij} g_1(x_j) \qquad \forall i \in \omega_h^{(r)}$$

mit:

- $g_1(x_j)$: Wert der Dirichlet Randbedingung im Punkt x_j
- $\omega_h^{(r)}$: Indexmenge, welche die Nummern der Nicht-Dirichlet-Knoten im Element $T^{(r)}$ enthält.
- $\gamma_h^{(r)}$: Indexmenge, welche die Nummern der Dirichlet-Knoten im Element $T^{(r)}$ enthält.

Wir könnten auch diese Korrekturen am Ende des Assemblierungsprozesses machen. Allerdings ist es bei der Skyline-Speichertechnik für die Matrix aufwendig, die Matrixeinträge der Zeilen zu finden, die genullt werden müssen.

Dann werden die Elementsteifigkeitsmatrizen und Elementlastvektoren durch den folgenden Algorithmus assembliert, der eine Schleife über die Elementbereiche macht, weil verschiedene Bereiche verschiedene Wärmeleitzahlen haben können.

Für jeden Elementbereich

Für jedes Element $T^{(r)}$ des Bereichs

Berechne $K^{(r)}$ und $\underline{f}^{(r)}$

Berücksichtigung der Randedingungen 1. Art: $\underline{f}^{(r)}$ und $K^{(r)}$ werden korrigiert

Für jeden Knoten des Elements, der i als globale Nummer und k als lokale Nummer hat.

$$f_i := f_i + f_k^{(r)}$$

Für Jeden Knoten des Elements, der j als globale Nummer und l als lokale Nummer hat.

$$K_{ij} := K_{ij} + K_{kl}^{(r)}$$

Der nächste Schritt ist die Berücksichtigung der Randbedingungen:

Randbedingungen 2. Art:

Vektor, der für die Neumannschen Randbedingungen assembliert werden muss:

$$\underline{f}^{(e_2)} = \left[\left(\int_0^1 \varphi_i(\xi) d\xi \right) g_2 \sqrt{\left(x_{n_2} - x_{n_1}\right)^2 + \left(y_{n_2} - y_{n_1}\right)^2} \right]_{i=1}^{\bar{N}_E}$$

mit:

- φ_i sind die über dem Referenzintervall definierten Formfunktionen
- $\bar{N}_E$: Anzahl der Knoten pro Kante
- n_1 und n_2 : Anfangs- und Endknote der Kante
- g_2 : Wert der Randbedingungen auf der Kante

Randbedingungen 3. Art:

Vektor, der für die Robinschen Randbedingungen assembliert werden muss:

$$\underline{f}^{(e_3)} = \left[\left(\int_0^1 \varphi_i(\xi) d\xi \right) \alpha \, u_A \sqrt{\left(x_{n_2} - x_{n_1}\right)^2 + \left(y_{n_2} - y_{n_1}\right)^2} \right]_{i=1}^{\bar{N}_E}$$

mit: α und u_A : Werte der Randbedingungen auf der Kante

Matrix, die für die Robinschen Randbedingungen assembliert werden muss:

$$K^{(e_3)} = \left[\left(\int_0^1 \varphi_i(\xi) \varphi_j(\xi) d\xi \right) \alpha \sqrt{\left(x_{n_2} - x_{n_1}\right)^2 + \left(y_{n_2} - y_{n_1}\right)^2} \right]_{i=1}^{\bar{N}_E}$$

Natürlich korrigiert das Programm auch den Vektor und die Matrix, die Robinschen Randbedingungen entsprechend, um die Randbedingungen 1. Art zu berücksichtigen. Sie werden analog wie die Elementsteifigkeitsmatrix und der Elementlastvektor korrigiert.

Diese Matrizen und Vektoren werden mit denselben Algorithmus wie die Elementsteifigkeitsmatrizen und Elementvektoren assembliert.

Randbedingungen 1. Art:

Da die Berücksichtigung der Randbedingungen 1. Art während der Berechnung der Steifigkeitsmatrix und des Lastvektors ohne Berücksichtigung der Randbedingungen und während der Einarbeitung der Robinschen Randbedingungen behandelt wird, muss jetzt nur noch „1“ auf die Diagonale der globalen Steifigkeitsmatrix und die Werte der Dirichlet-Knoten auf den Lastvektor gespeichert werden.

Für jeden Dirichlet-Knoten des Elements, der i als globale Nummer hat,

$K_{ii} = 1$

$f_i = g_1(x_i)$

mit $g_1(x_i)$: Wert der Dirichletschen Randbedingung im Punkt x_i.

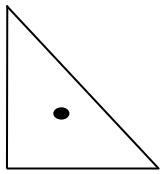
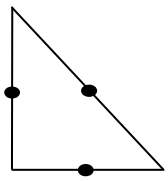
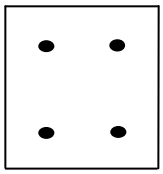
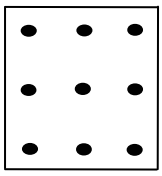
2.4.2. Numerische Integration

Bei der Berechnung der Einträge der Matrizen und Vektoren ist es nicht immer möglich, die entsprechenden Integrale analytisch zu berechnen. Deshalb nutzt das Programm Quadraturformeln zur Berechnung der Integrale. Die Grundidee der Quadraturformeln besteht darin, die Integrale durch Summen zu approximieren. Die Software benutzt die Quadratur der Gestalt:

$$\int_{\bar{T}} w(\xi)d\xi \approx \sum_{i=1}^{l} \alpha_i w(\xi_i)$$

Da die Polynome, die das Programm integrieren muss, immer von relativ kleinem Grad sind, kann FEJavaDemo Quadraturstützstellen benutzen, um diese exakt zu integrieren:

Lage der Stützstellen	Quadraturstützstellen ξ_i oder $(\xi_{i,1}, \xi_{i,2})$	Gewichte α_i	exakt für Polynome vom Grad k
	$\frac{3-\sqrt{3}}{6}, \frac{3+\sqrt{3}}{6}$	$\frac{1}{2}, \frac{1}{2}$	k = 3
	$\frac{5-\sqrt{15}}{10}, \frac{1}{2}, \frac{5+\sqrt{15}}{10}$	$\frac{5}{18}, \frac{8}{18}, \frac{5}{18}$	k = 5

	$\left(\frac{1}{3},\frac{1}{3}\right)$	$\frac{1}{2}$	k = 1
	$\left(\frac{1}{2},0\right),\left(\frac{1}{2},\frac{1}{2}\right),\left(0,\frac{1}{2}\right)$	$\frac{1}{6},\frac{1}{6},\frac{1}{6}$	k = 2
	$\left(\frac{3-\sqrt{3}}{6},\frac{3+\sqrt{3}}{6}\right),\left(\frac{3+\sqrt{3}}{6},\frac{3+\sqrt{3}}{6}\right)$ $\left(\frac{3-\sqrt{3}}{6},\frac{3-\sqrt{3}}{6}\right),\left(\frac{3+\sqrt{3}}{6},\frac{3-\sqrt{3}}{6}\right)$	$\frac{1}{4},\frac{1}{4}$ $\frac{1}{4},\frac{1}{4}$	k = 3
	$\left(\frac{5-\sqrt{15}}{10},\frac{5+\sqrt{15}}{10}\right)\left(\frac{1}{2},\frac{5+\sqrt{15}}{10}\right)\left(\frac{5+\sqrt{15}}{10},\frac{5+\sqrt{15}}{10}\right)$ $\left(\frac{5-\sqrt{15}}{10},\frac{1}{2}\right)\left(\frac{1}{2},\frac{1}{2}\right)\left(\frac{5+\sqrt{15}}{10},\frac{1}{2}\right)$ $\left(\frac{5-\sqrt{15}}{10},\frac{5-\sqrt{15}}{10}\right)\left(\frac{1}{2},\frac{5-\sqrt{15}}{10}\right)\left(\frac{5+\sqrt{15}}{10},\frac{5-\sqrt{15}}{10}\right)$	$\frac{25}{324},\frac{10}{81},\frac{25}{324}$ $\frac{10}{81},\frac{16}{81},\frac{10}{81}$ $\frac{25}{324},\frac{10}{81},\frac{25}{324}$	k = 5

Quadraturformeln über dem Referenzintervall I = [0,1] und über dem Referenzelement

Bemerkung: Bei Vierecken sind die Formeln exakt für Funktionen, die Polynome k-ten Grades in jeder Raumrichtung sind.

Die numerische Integration wird mehrfach genutzt:

- Bei der Berechnung der Elementsteifigkeitsmatrizen und der Elementlastvektoren. Das Programm führt in den Integralen eine Variablensubstitution durch, so dass Integrale nur über dem Referenzelement (Dreieck, Viereck, usw.) zu berechnen sind.
- Bei der Einarbeitung der Randbedingungen, rechnet FEJavaDemo auf analoge Weise nur über dem Referenzintervall I = [0,1].

2.5. Lösung des FE-Gleichungssystems

Als Resultat der FE-Diskretisierung haben wir ein lineares Gleichungssystem: $K\,u = f$.
K ist die Steifigkeitsmatrix, f der Lastvektor und u der Lösungsvektor, den wir suchen.

Zur Lösung der FE-Gleichungssysteme kann man sowohl direkte als auch iterative Verfahren benutzen. Direkte Auflösungsverfahren sind Algorithmen, die den Lösungsvektor u in endlich vielen Schritten liefern. Iterative Verfahren bestimmen den Vektor u ausgehend von einer Startnäherung u_0 als Grenzwert einer Folge von Näherungslösungen (u_k).

Wir wissen, dass die Matrix K symmetrisch und positiv definiert ist und in diesem Fall wird häufig das *Cholesky-Verfahren* eingesetzt, das ein direktes Verfahren ist.

Beim Cholesky-Verfahren wird die Matrix K in das Produkt einer oberen Dreiecksmatrix und ihrer transponierte Matrix zerlegt: $K = S^T S$

Wir wissen auch, dass die Matrix K schwach besetzt ist. Bei der Durchführung der Faktorisierung bleiben gewisse Besetztheitsstrukturen der Matrix K erhalten. So kann man die Matrizen K und S auf den gleichen Speicherplätzen abspeichern. Und vor allem kann das Programm einen besonderen Zerlegungsalgorithmus benutzen, der die Nulleinträge außerhalb der Hülle nicht berechnet:

Zerlegungsalgorithmus ($K = S^T S$) mit Berücksichtigung der Profilstruktur von K:

$$S_{11} = \sqrt{K_{11}}$$

Berechne für $j = 2, 3, \ldots, N$

Falls $l_0(j)+1 < j$, berechne für i = $l_0(j)+1, l_0(j)+2, \ldots, j - 1$:

$$S_{ij} = \frac{1}{S_{ii}}\left(K_{ij} - \sum_{l=\max\{l_0(i),l_0(j)\}+1}^{i-1} S_{li} S_{lj} \right)$$

$$S_{jj} = \sqrt{K_{jj} - \sum_{l=l_0(j)+1}^{j-1} S_{lj}^2}$$

(Dabei bezeichnet $l_0(j)$ den Zeilenindex, für den in der j-ten Spalte $K_{ij} = 0$ für alle $0 < i < l_0(j) + 1$ und $K_{l_0(j)+1,\,j} \neq 0$ gilt)

Die Lösung von $Ku = f$ ist somit der Lösung von $S^T S\,u = f$ äquivalent. Dieses Gleichungssystem löst das Programm in zwei Etappen durch *Vorwärts*- und *Rückwärtseinsetzen*, d.h. es löst zunächst

$S^T\ y = f$ und dann $S\ u = y$. Die Systemmatrizen dieser zwei Gleichungssysteme haben eine Dreiecksgestalt. Derartige Gleichungssysteme sind besonders einfach lösbar:

Algorithmus des Vorwärtseinsetzen:

$$\begin{pmatrix} S_{11} & 0 & \cdots & 0 \\ S_{12} & S_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ S_{1N} & S_{2N} & \cdots & S_{NN} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}$$

$$y_1 = \frac{f_1}{S_{11}}$$

$$y_i = \frac{1}{S_{ii}} \left(f_i - \sum_{j=1}^{i-1} S_{ji} y_j \right) \qquad i = 2,\ 3,\ \ldots,\ N$$

Algorithmus des Rückwärtseinsetzen:

$$\begin{pmatrix} S_{11} & S_{12} & \cdots & S_{1N} \\ 0 & S_{22} & \cdots & S_{2N} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & S_{NN} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$u_N = \frac{y_N}{S_{NN}}$$

$$u_i = \frac{1}{S_{ii}} \left(y_i - \sum_{j=i+1}^{N} S_{ij} u_j \right) \qquad i = N-1,\ N-2,\ \ldots,\ 1$$

3. Struktur der Eingabefiles

Um eine Vernetzung bereitzustellen, muss man Datenfiles schreiben: ein „.net“ File und ein „.dat“ File. Das erste File enthält Informationen über die Vernetzung und der zweite über die Randbedingungen, die Wärmeleitzahlen und die Funktion *f*.

3.1. Das .net File

Linien, die mit # beginnen, sind Kommentar und das Programm liest sie nicht.

In der ersten Zeile wird die Art der Elemente angegeben (1 für Dreiecke, 2 für Vierecke usw.).

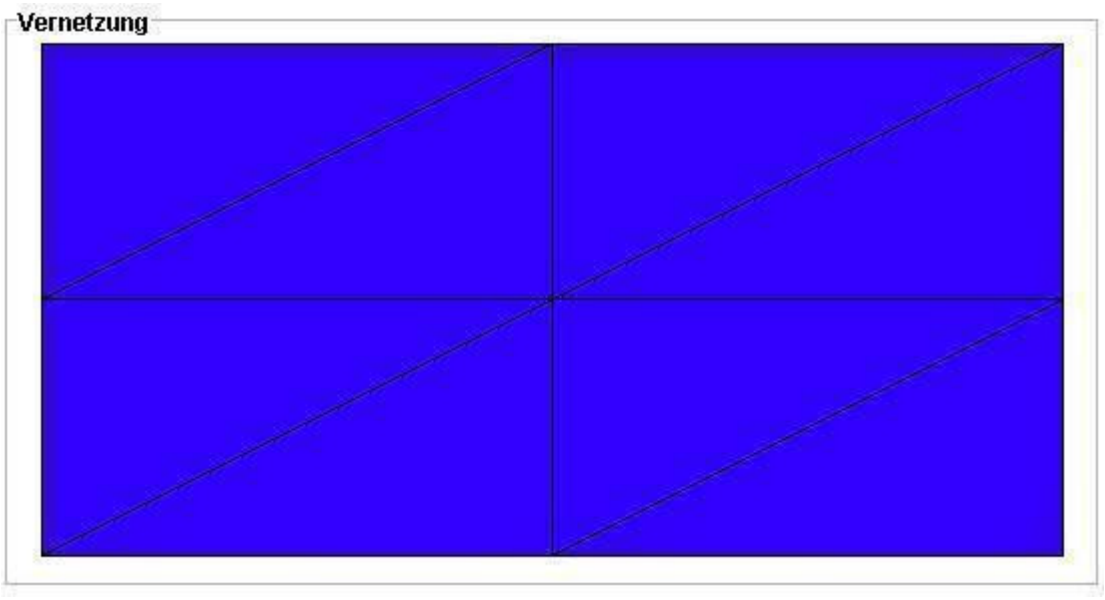
Dann gibt es die Anzahl die Knoten (d.h. die Dimension der globalen Steifigkeitsmatrix) und die Anzahl der Elemente.

Für die Definition der Knoten müssen drei Zahlen angegeben werden: die globale Nummer, die X-Koordinate und die Y-Koordinate.

Dann werden die Elemente mit ihrer globalen Nummer, den globalen Nummern ihrer Eckpunkte (drei Nummern für Dreiecke, vier für Vierecke, ...) und die Materialbereichsnummer definiert.

Die nächste Zeile enthält die Anzahl der Randkanten und dann folgen die Definitionen der Randkanten. Sie werden mit ihrer globalen Nummer und den zugehörigen Anfangs- und Endknoten definiert.

Das folgende Beispiel ist das .net File einer einfachen Vernetzung



```
# FEJavaDemo - Netzfile
#
# Art der Elemente (1 für Dreiecke, 2 für Vierecke usw.)
  1
#
# Anzahl der Knoten und Anzahl der Elemente
  9 8
#
# Globale Nummer und Knotenkoordinaten
  1  0.0  0.0
  2  0.0  1.25
  3  0.0  2.5
  4  2.5  0.0
  5  2.5  1.25
  6  2.5  2.5
  7  5.0  0.0
  8  5.0  1.25
  9  5.0  2.5
#
# Globale Nummern der Knoten jeder Elemente (und Nummer des MaterialBereich)
  1  4  5  1  1
  2  2  1  5  1
  3  5  6  2  1
  4  3  2  6  1
  5  7  8  4  1
  6  5  4  8  1
  7  8  9  5  1
  8  6  5  9  1
#
# Anzahl der Randkanten
  8
#
# Anfangs- und Endknoten der Randkanten (und globale Nummer)
  1   1  4
  2   4  7
  3   7  8
  4   8  9
  5   9  6
  6   6  3
  7   3  2
  8   2  1
```

3.2. Das .dat File

Am Beginn des Files werden die Anzahl der Materialbereiche und dann die Wärmeleitzahlen λ_1 und λ_2 in jedem Bereich geschrieben.

Dann werden die Randbereiche definiert:

Zuerst wird die Anzahl der verschiedenen Randbereiche geschrieben.

Für jeden Bereich sind zwei Zahlen einzugeben: die Anzahl der Kanten und die Art der Randbedingung (1 für Dirichlet, 2 für Neumann und 3 für Robin).

Nun kommen die globalen Nummer der Kante jeder Randbereich und die zugehörigen Randdaten.

- Für die Dirichletschen Randbedingungen muss man zwei Werte angeben, die die Temperatur im Anfangs- und Endknoten sind.
- Was die Neumannschen Randbedingungen betrifft, wird nur ein Wert eingegeben, der der Wärmefluß auf der Kante darstellt $\left(\frac{\partial u}{\partial N} = a \right)$
- Und für die Robinschen Randbedingungen muss man zwei Werte angeben: *a* und *b* $\left(\frac{\partial u}{\partial N} = a \left[b - u(x) \right] \right)$

Die letzte Information dieser Datei ist die Funktion *f*. Sie kann verschieden für jeden Bereich sein und wird als Funktionsausdruck geschrieben (z.B. 2*x+tan(y)+7). Die Funktionen, die genutzt werden können, sind die folgende: tan(), sin(), cos(), exp(), ln(), sqrt() und die Konstante π.

Das folgende Beispiel ist das .dat File, das dem ersten Beispiel entspricht:

```
# FEDemoJava - .dat Datei
#
# Anzahl der Materialbereiche
  2
# Lambda1 und Lambda 2 in MaterialBereich 1
  200.0  200.0
# Lambda1 und Lambda 2 in MaterialBereich 2
  100.0  100.0
#
# Anzahl der verschiedenen Randbereiche
  3
# Anzahl der Kanten und Art der Randbedingung in Randbereich 1
  4  1
# Anzahl der Kanten und Art der Randbedingung in Randbereich 2
  2  2
# Anzahl der Kanten und Art der Randbedingung in Randbereich 3
  2  3
#
# Globale Nummer und Randdaten (a (und b) Koeff) der Kante im Randbereich 1
  1   0.   10.
  2  10.   20.
  3  20.   30.
  4  30.   40.
# Globale Nummer und Randdaten (a (und b) Koeff) der Kante im Randbereich 2
  5   5.
  6   7.
# Globale Nummer und Randdaten (a (und b) Koeff) der Kante im Randbereich 3
  7  10.   50.
  8  10.   50.
#
# Funktion F(x,y) in MaterialBereich 1
  3*cos(x*y*Pi)
# Funktion F(x,y) in MaterialBereich 2
  0
```

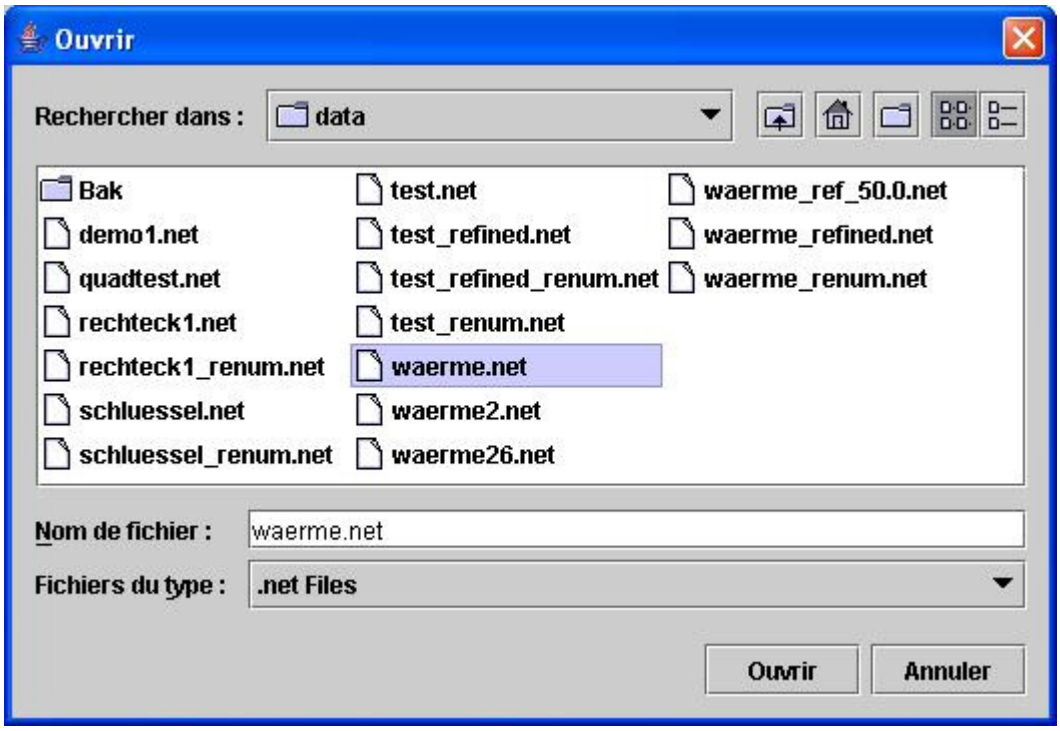
4. Programmablauf

Wahl der Art der Ansatzfunktionen:

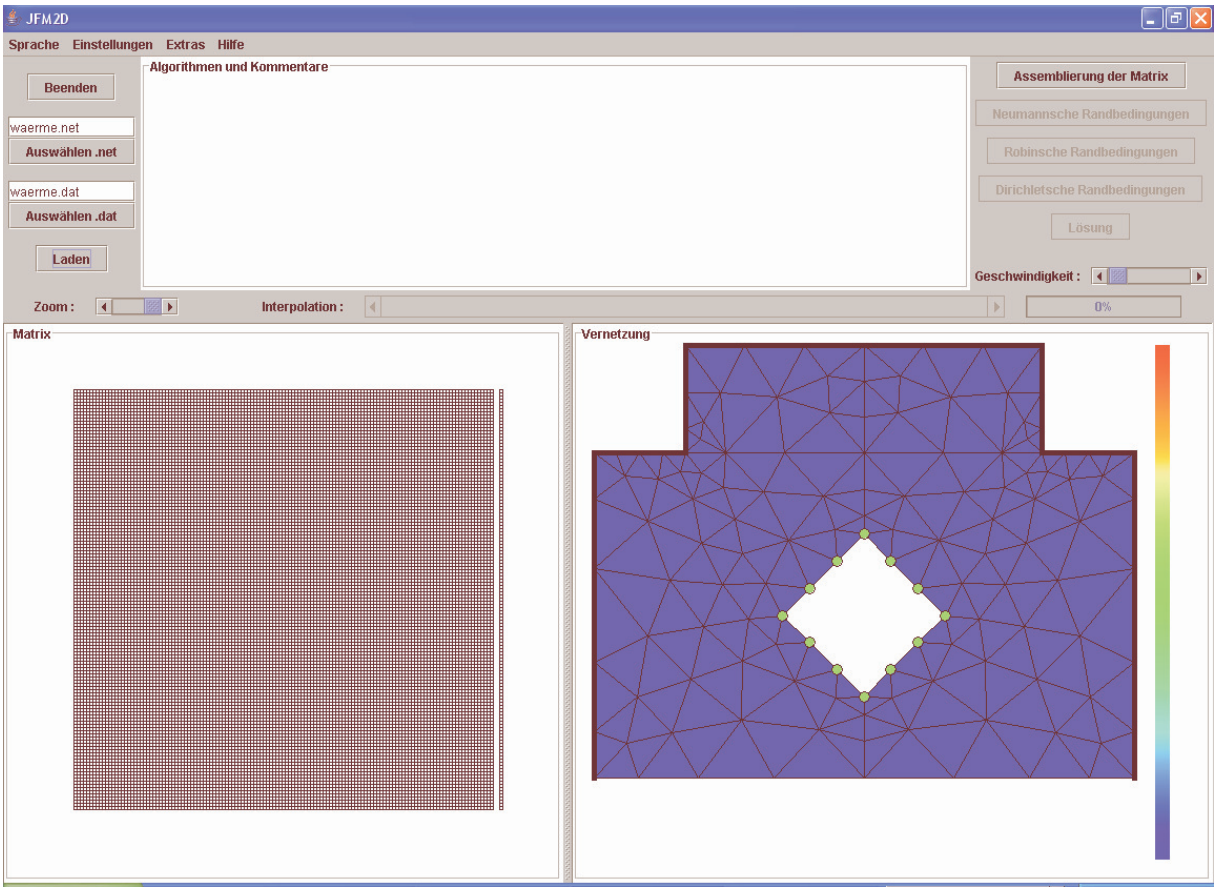
Standardmäßig arbeitet das Programm mit linearen Ansatzfunktionen. Durch den Menüpunkt „Einstellungen → Art der Ansatzfunktionen“ können auch quadratische Ansatzfunktionen gewählt werden.

Einlesen der benötigten Datenfiles:

Durch Anklicken des Knopfs „Auswählen .net“ (bzw. „Auswählen .dat“) werden die zur Verfügung stehenden Netzfiles (bzw. „Dat“-files) angezeigt.



Nach Auswahl den gewünschten Files mittels „Laden“ werden diese geladen.



Aufbau der Steifigkeitsmatrix und des Lastvektors ohne Berücksichtigung der Randbedingungen:

Sobald das Einlesen des Netz- und Datenfiles beendet ist, wird der Knopf „Assemblierung der Matrix“ aktiv. Er dient dazu, die Berechnung und Assemblierung der Steifigkeitsmatrix und des Lastvektors ohne Randbedingungen zu starten.

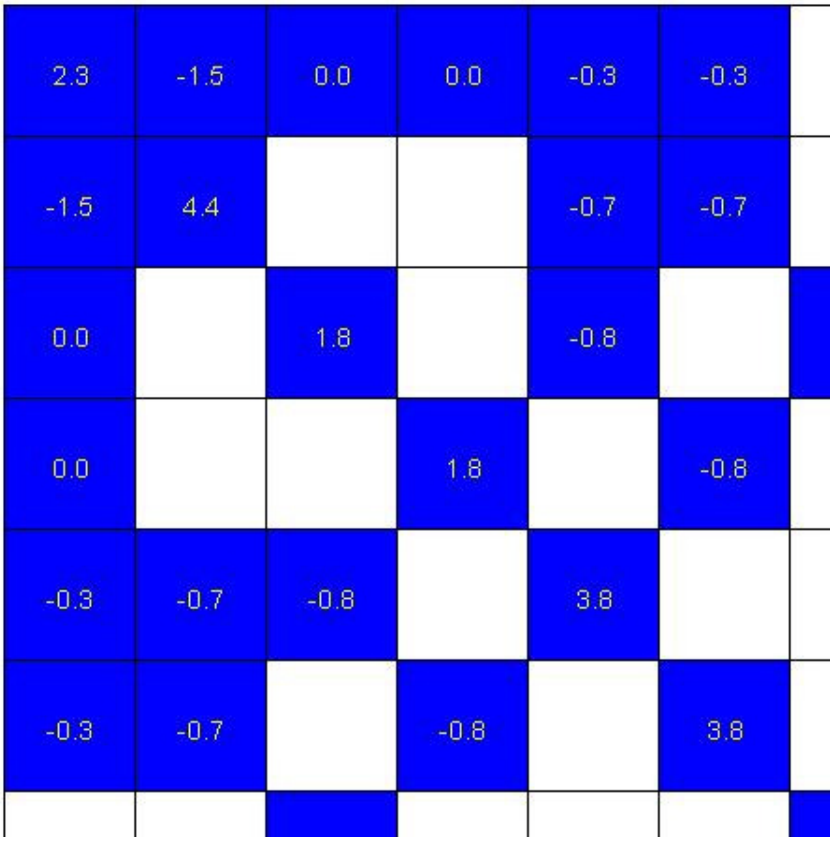
Während dieses Prozesses, erlaubt ein Scrollbar, die Geschwindigkeit der Anzeige zu verändern. Oft zieht man vor, dass das Programm schnell die Lösung liefert. Deshalb werden die Stufen der Rechnung nicht mehr angezeigt, wenn der Rollbalken am weitesten links ist.

Berücksichtigung der Randbedingungen:

Durch Anklicken der drei folgenden Knöpfe werden die Randbedingungen berücksichtigt.

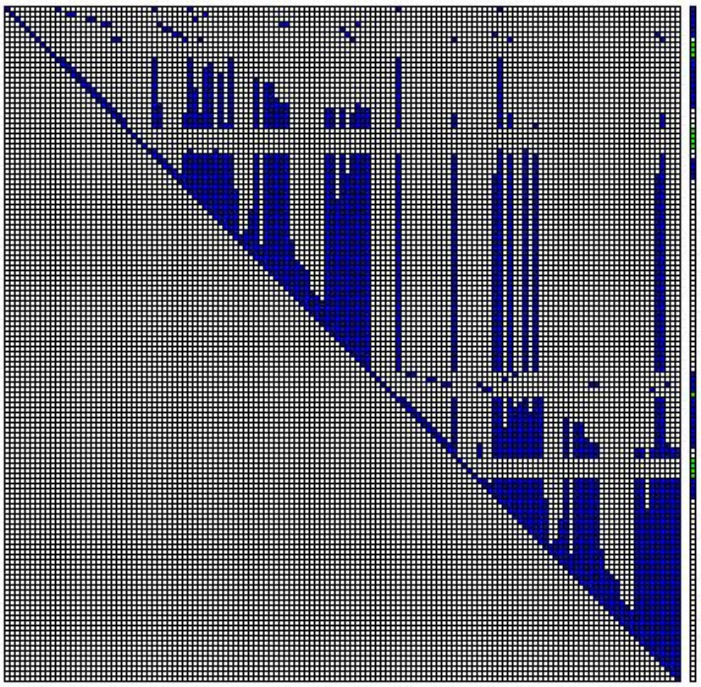
Während der verschiedenen Assemblierungsprozesse, sind die Elemente der Vernetzung und die jeweiligen Matrixeinträge, mit denen das Programm arbeitet, in Rot eingefärbt. Die Nicht-Null-Einträge sind blau, um hervorzuheben, dass die Matrix schwach besetzt ist. Die grünen Elemente entsprechen den Dirichlet-Knoten.

Ein Scrollbar erlaubt, den Zoom auf die Matrix zu verändern. Wenn der Rollbalken am weitesten links ist, werden die Werte der Einträge angezeigt. Zwei Leisten deuten die Nummern der Spalten und Zeilen an.

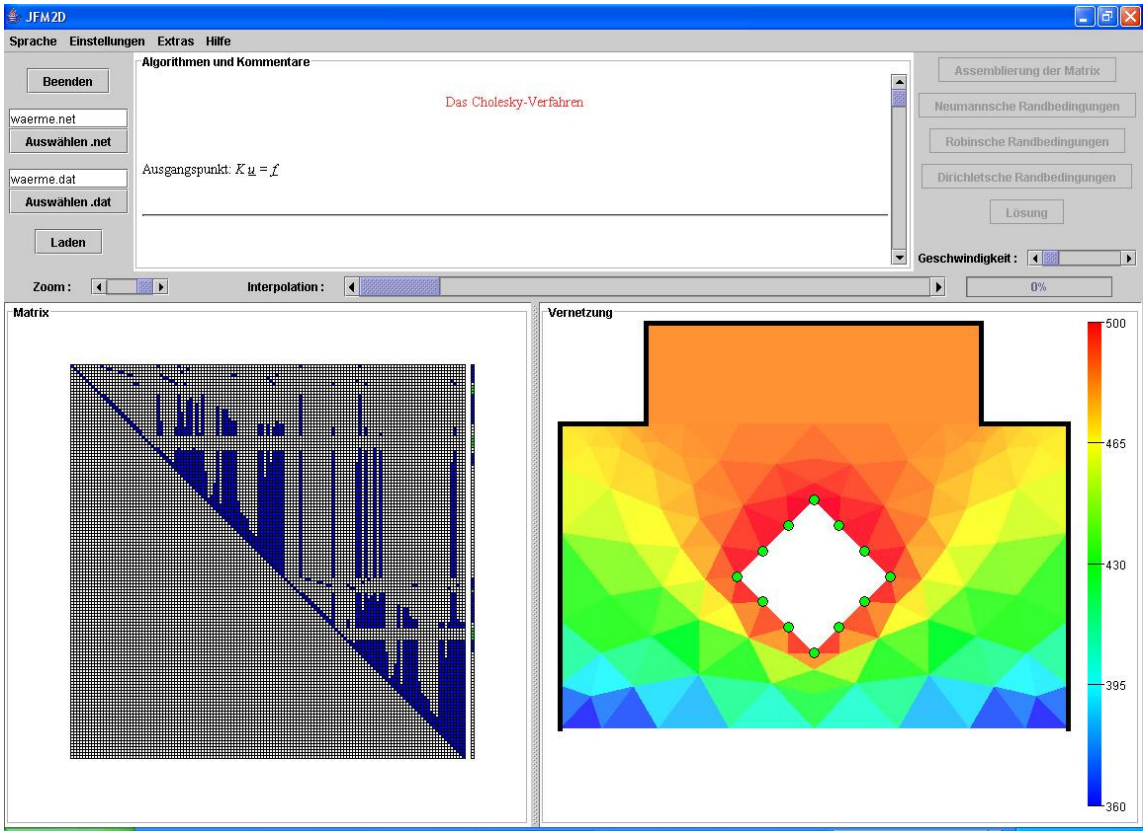


Lösung des FE-Gleichungssystems:

Nachdem alle Randbedingungen berücksichtigt wurden, leitet der Knopf „Lösung“ den Auflösungsprozess ein. Mittels Auswahl des Menüpunktes „Einstellungen → Anzeige → S'S Zerlegung zeigen“, wird zuerst die obere Dreiecksmatrix S der S^TS Zerlegung angezeigt. Ein kleiner Zeichentrick symbolisiert die Durchführung des Vorwärts- und Rückwärtseinsetzens.

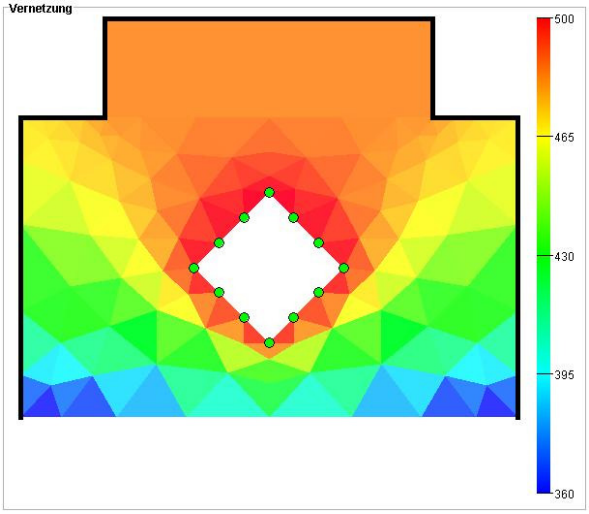


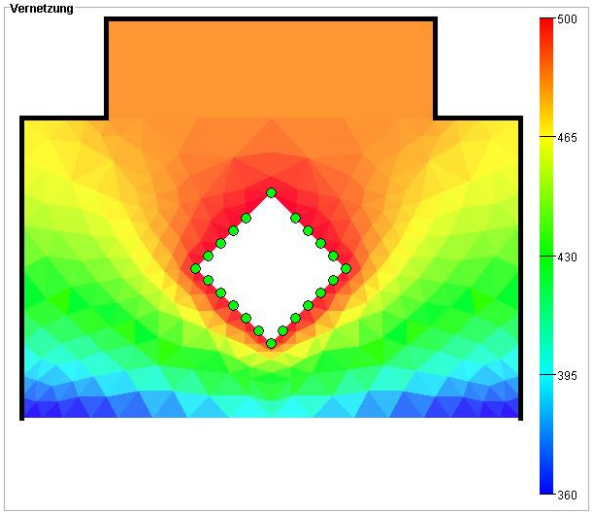
Nachdem die Auflösung beendet ist, werden das Temperaturfeld und eine Temperaturskala dargestellt.



Verfeinerung der Vernetzung:

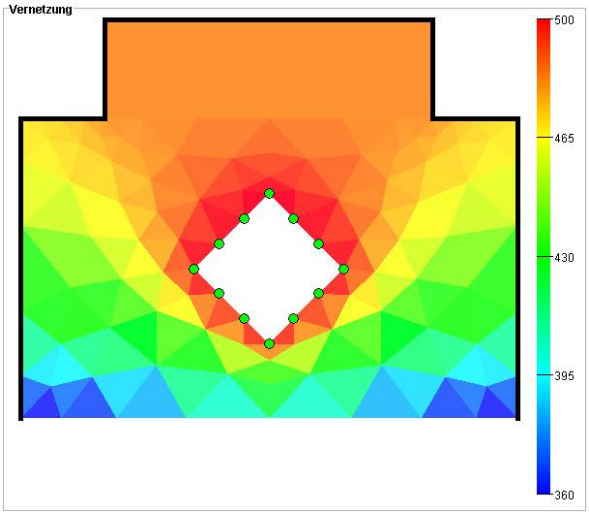
Falls das Programm mit linearen Ansatzfunktionen und Dreiecken arbeitet, kann auch die Vernetzung verfeinert werden („Extras → Verfeinerung der Vernetzung“). Wenn eine Verfeinerung gefordert wird, wird eine höchste zugelassene Temperaturabweichung zwischen zwei Eckpunkte abgefragt. Falls „0“ eingegeben wurde, wird die Vernetzung vollständig verfeinert aber nur ein Mal. Es gibt die Möglichkeit, die Dateien der neuen Vernetzung zu erstellen („Extras → Neue Dateien erstellen ... → ... für Verfeinerung“).

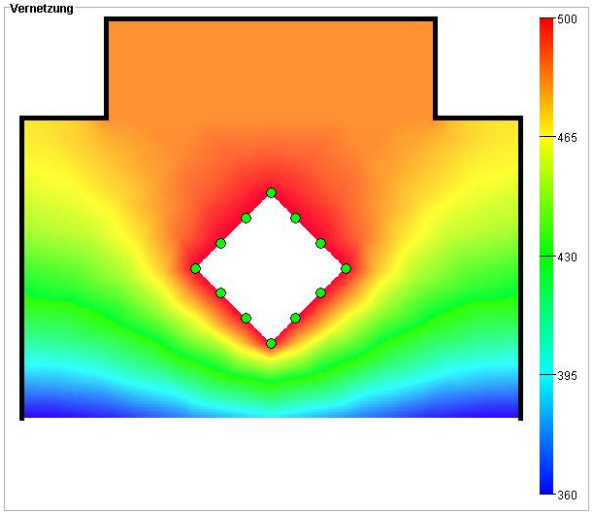




Interpolation der Lösung:

Durch den Scrollbar „Interpolation“ wird die Lösungsdarstellung verbessert: die Elemente können intern wiederholt zerlegt werden und die Lösung wird über den neuen Knoten linear interpoliert. Da es ziemlich lang dauern kann, gibt es einen Fortschrittsbalken, der dazu dient, ein Feedback über den aktuellen Status der Arbeit zu haben.



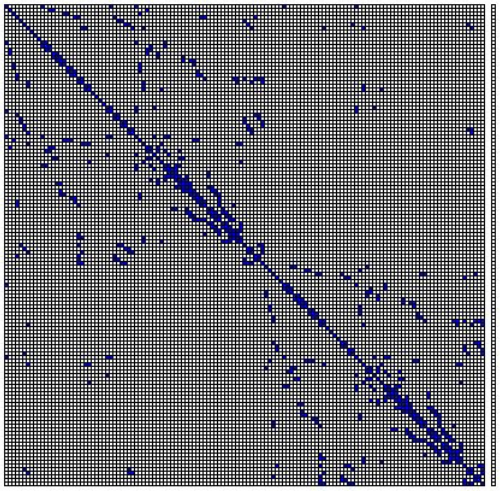


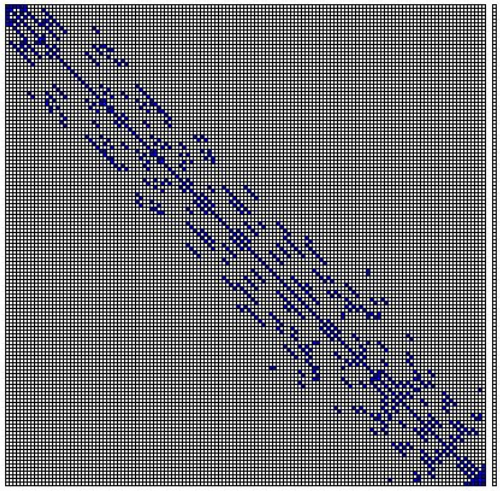
Speicherung der Lösung:

Die Lösung kann (d.h. die Knoten mit Ihren Koordinaten und dem Temperaturwert) mittels „Extras → Lösung speichern“ Menüeintrag gespeichert werden.

Umnummerierung der Knoten:

Die Knoten können auch durch das „Extras → Knotenumnummerierung“ Menü umnummeriert werden. Der Menüeintrag „Automatische Knotenumnummerierung bei quadratischen Funktionen“ ist standardmäßig ausgewählt. Er erlaubt den Prozess der Auflösung des Problems schneller zu machen. Es gibt die Möglichkeit, die Dateien der neuen Vernetzung zu erstellen („Extras → Neue Dateien erstellen ... → ... für Umnummerierung“).





Andere Möglichkeiten:

Natürlich kann die Software dank dem „Sprache“ Menü übersetzt werden.

Zwischen jedem Schritt können Jpeg-Files erstellt werden, um Bilder der Matrix und der Vernetzung zu erhalten („Extras → Jpeg-Files erstellen“).

Dank dem „Einstellungen → Anzeige“ Menü, ist es durch den „Randbedingungen zeigen“ Menüeintrag möglich, die Dirichlet-Knoten in grün einzufärben und die wärmeisolierten Kanten zu betonen. Dazu können die Knotennummern angezeigt werden („Einstellungen → Anzeige → Knotennummern anzeigen“).