Technische Universität Chemnitz-Zwickau

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Matthias Pester

Behandlung gekrümmter Oberflächen in einem 3D-FEM-Programm für Parallelrechner

Preprint SFB393/97-10

Internet-Version ohne Abbildungen

Preprint-Reihe des Chemnitzer SFB 393

SFB393/97-10 **April 1997**

Inhalt

Matthias Pester
TU Chemnitz-Zwickau
Fakultät für Mathematik
D-09107 Chemnitz

`pester@mathematik.tu-chemnitz.de`
`http://www.tu-chemnitz.de/~pester/`

1 Einleitung

Im folgenden wird eine einfache und zugleich für viele Anwendungen ausreichende Methode zur Beschreibung und Verarbeitung gekrümmter Flächen in einem 3D-Finite-Elemente-Programm ([1], [10], [12], [13]) vorgestellt.

Die mehr oder weniger automatische Netzgenerierung für 3D-Rechnungen, insbesondere deren Parallelisierung, ist ein aktueller Forschungsgegenstand, für den sich verschiedene Zugänge anbieten ([6], [7], [9]). Unsere prinzipielle Vorgehensweise besteht darin, die geometrischen Eigenschaften einer Fläche in kompakter Form zu beschreiben und bei der Netzverfeinerung in geeigneter Weise mit einzubeziehen.

Obwohl nicht auf Parallelrechner beschränkt, ist die hier beschriebene Methode dort von besonderer Bedeutung. Ein wesentliches Merkmal bei der Entwicklung paralleler Algorithmen ist das Bestreben nach durchgängiger Parallelität bei einer möglichst balancierten Verteilung der Last auf die Prozessoren, das betrifft Speicher- und Rechenkapazität gleichermaßen. Kommunikationen zwischen den Prozessoren sind bis auf das notwendige Minimum zu vermeiden. Das wird durch eine möglichst grobe Definition des Berechnungsgebietes auf der Basis einer Geometriebeschreibung erreicht. In der vorliegenden Realisierung bestimmt dieses *Grobnetz* durch die Anzahl der Volumenelemente zugleich den initialen Aufwand zur (sequentiellen) Datenbereitstellung und den anfänglichen Grad der Parallelisierung, indem zunächst nur ganze Grobnetzelemente einzelnen Prozessoren zugeordnet sind. Aus dieser Verteilung der Elemente resultiert auch im wesentlichen der Kommunikationsaufwand bei der Lösung des Gleichungssystems ([2], [10]). Durch Berücksichtigung der geometrischen Gestalt von Flächen bei der Netzverfeinerung ist es möglich, das Startnetz relativ grob vorzugeben und damit die anfängliche Verteilung des Netzes auf die Prozessoren des Parallelrechners auf diesem groben Level mit geringem Kommunikationsaufwand vorzunehmen. Die dann vollständig parallel ablaufende Netzgenerierung liefert so mit wachsendem Level der Verfeinerung eine bessere Approximation der gekrümmten Flächen.

An den Untersuchungen zu den in Abschnitt 4 genannten Beispielen waren auch beteiligt: Arnd Meyer, Uwe Reichel, Thomas Grund und Kornelia Pietsch. Die Problematik der Vererbung von Geometriedaten bei der Netzverfeinerung wurde zu einem Teil von Frank Milde realisiert. Als Hilfsmittel zur geometrischen Transformation vorhandener Netzdaten in Standardfiles kann ein Programm von Dag Lohse verwendet werden. Die in den Abbildungen verwendeten Netzdarstellungen wurden zum Teil mit den im erwähnten 3D-FEM-Programm integrierten eigenen Visualisierungstools ([12]), zum Teil mit einem externen Visualisierungsprogramm auf der Basis von GRAPE ([14]) erzeugt.

2 Geometriedefinition im Grobnetz

Die Beschreibung des gesamten Grobnetzes für das Programm *SPC-PM Po 3D* erfolgt mittels standardisiertem Eingabe-File in weitgehend freiem und lesbarem Format. Die genaue Struktur ist in [1] definiert. In diesem File werden unter anderem die topologischen Zusammenhänge des dreidimensionalen Gebietes unter den Schlüsselworten `#VERTEX, #EDGE, #FACE, #SOLID` als Hierarchie von k-dimensionalen Mannigfaltigkeiten definiert (k-Faces, $k = 0, 1, 2, 3$). Dabei folgt jeweils dem *Namen* eines k-Faces ($k > 0$) eine *Typ*angabe und seine Definition durch eine Anzahl von $(k-1)$-Faces. Abbildung 1 zeigt ein Beispiel.

```
   ...
#VERTEX: 6
   1    0.0  -1.0   0.0
   2    1.0   0.0   0.0
   3    0.0   1.0   0.0
   4   -1.0   0.0   0.0
   5    0.0   0.0   1.0
   6    0.0   0.0   0.0
#EDGE: 13
   12     0     1  2
   23     0     2  3
   34     0     3  4
   41     0     4  1
   15     0     1  5
   25     0     2  5
   35     0     3  5
   45     0     4  5
   16     0     1  6
   26     0     2  6
   36     0     3  6
   46     0     4  6
   56     0     5  6
#FACE: 12
    1     0     3     12  25  15
    2     0     3     23  35  25
    3     0     3     35  45  34
    4     0     3     41  15  45
    5     0     3     12  26  16
    6     0     3     23  36  26
    7     0     3     34  46  36
    8     0     3     14  16  46
    9     0     3     16  15  56
   10     0     3     26  25  56
   11     0     3     36  35  56
   12     0     3     46  45  56
#SOLID: 4
      1   1     4     1   5   9  10
      2   1     4     2   6  10  11
      3   1     4     3   7  11  12
      4   1     4     4   8  12   9
   ...
```

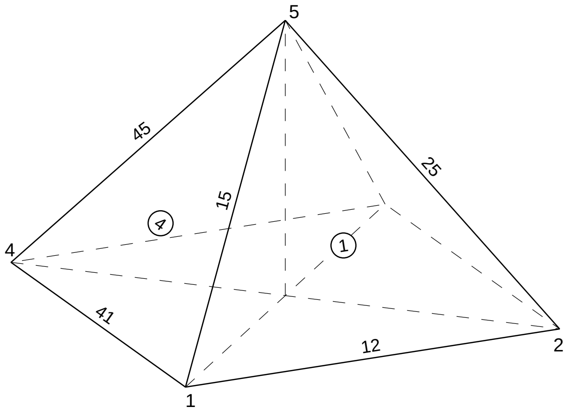


Abbildung 1: Beispiel zur hierarchischen Definition der Gebietstopologie: aus vier Tetraedern zusammengesetzte quadratische Pyramide

```
  ...
#FACE: 12
    1    2    3    12  25  15
    2    2    3    23  35  25
    3    2    3    35  45  34
    4    2    3    41  15  45
    5    0    3    12  26  16
    6    0    3    23  36  26
    7    0    3    34  46  36
    8    0    3    14  16  46
    9    0    3    16  15  56
   10    0    3    26  25  56
   11    0    3    36  35  56
   12    0    3    46  45  56
  ...
#FACE_GEO: 1
    2   31   7   0. 0. 1.  0. 0. 0.  1.
  ...
```

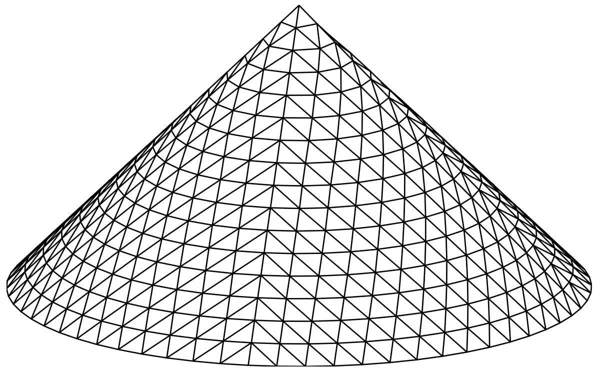


Abbildung 2: Modifikationen zur Geometriebeschreibung, das Netz aus Abb. 1 nach viermaliger Verfeinerung

Besondere geometrische Eigenschaften können somit einer Fläche (2-Face) einfach über die vorgesehene Typangabe zugeordnet werden. Diese Angabe ist zu verstehen als Zeiger auf eine detaillierte Charakteristik der Flächengeometrie. Wir beschränken uns dabei zunächst auf einfache, mit wenigen Parametern zu beschreibende Formen (Zylinder, Kugel, Kegel, Rotationskörper, Torus, ...). Dazu enthält das Eingabe-File unter dem Schlüsselwort `#FACE_GEO` für jede unterschiedliche Flächengeometrie jeweils diese notwendigen Parameter. Im betrachteten Beispiel aus Abb. 1 genügt die Angabe nur einer (gemeinsamen) Geometrie für alle vier Seitenflächen der Pyramide, um daraus einen Kegel zu generieren. Die notwendigen Änderungen zeigt Abb. 2. Die vier Seitenflächen erhalten als `<typ>` einen Eintrag `"2"`, der auf eine mit `"2"` beginnende Zeile im Abschnitt `FACE_GEO` verweist.

Die Eintragungen in der Geometriebeschreibung folgen dem gleichen Prinzip wie die Definition der anderen Objekte. Im Beispiel bedeutet die Datenzeile

```
  2  31  7    0. 0. 1.   0. 0. 0.   1.
```

in der angegebenen Reihenfolge:

`2`	Name der Geometrie(zeile), als Referenz in der Face-Definition verwendet
`31`	Codierung des Flächentyps, hier: Kegel
`7`	Anzahl der Parameter zur Definition der Fläche
`0. 0. 1.`	Koordinaten der Kegelspitze
`0. 0. 0.`	Koordinaten eines weiteren Punktes auf der Rotationsachse
`1.`	Radius des Kegels an der Stelle des zweiten Achsenpunktes

Die Geometriebeschreibung hat damit die allgemeine Form

```
<name>  <typ>  <n>  <parameter_1> ... <parameter_n>
```

wobei auf `<name>` in der Face-Definition Bezug genommen wird. Als `<typ>` wurden bislang in Zehnerschritten verschiedene geometrische Flächen definiert

`1 ... 9`	für eine Ebene
`11 ... 19`	für eine Zylinderfläche
`21 ... 29`	für eine Kugeloberfläche
`31 ... 39`	für einen Kegelmantel
`41 ... 49`	für eine Rotationsfläche (Ellipsoid, Hyperboloid)
`51 ... 59`	für eine Torusfläche

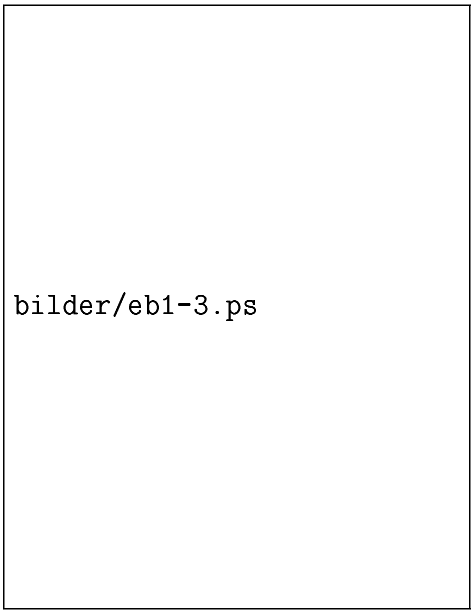
Die weitere Unterteilung jeweils von 1 bis 9 läßt lediglich die Möglichkeit offen, einen dieser Flächentypen auch auf verschiedene Art und Weise zu definieren, beispielsweise könnte die Kugel, wie von uns vorgesehen, mittels `<typ>=21` durch Mittelpunkt und Radius oder auch – falls es ein Nutzer so wünscht – mittels `<typ>=22` durch Mittelpunkt und einen Punkt auf der Oberfläche definiert sein, oder die Zylinderachse durch Punkt und Richtung statt zweier Punkte.

Für jeden der genannten Flächentypen existieren zur Zeit bereits Spezifikationen, die in Tabelle 1 zusammengestellt sind.

`<typ>`	`<n>`	Bedeutung der Parameter
1	6	$(n_x, n_y, n_z), (x_0, y_0, z_0)$ Normalenvektor und ein Punkt der Ebene
2	6	$(x_0, y_0, z_0), (n_x, n_y, n_z)$ Punkt der Ebene und Normalenvektor
11	7	$(x_1, y_1, z_1), (x_2, y_2, z_2), r$ zwei Punkte auf der Zylinderachse und Radius
21	4	$(x_m, y_m, z_m), r$ Mittelpunkt und Radius der Kugel
31	7	$(x_1, y_1, z_1), (x_2, y_2, z_2), r_2$ Kegelspitze, Punkt auf der Achse und Radius an dieser Stelle
41	8	$(x_1, y_1, z_1), (x_2, y_2, z_2), A, B$ zwei Punkte auf der Rotationsachse, wobei der erste Punkt als Ursprung eines lokalen (ξ, η)-Koordinatensystems betrachtet wird, sowie die Parameter der Kegelschnittgleichung der rotierenden Kurve: $\pm\frac{\xi^2}{a^2} \pm \frac{\eta^2}{b^2} = 1, \quad \text{mit } A = \pm a^2, B = \pm b^2$ Bem.: $B < 0 \;\rightarrow$ einschaliges, $A < 0 \rightarrow$ zweischaliges Hyperboloid
51	8	$(x_1, y_1, z_1), (n_x, n_y, n_z), r_1, r_2$ Mittelpunkt des Torus, Normalenvektor der Grundebene, Rotationsradius (Abstand des Mittelpunktes des rotierenden Kreises) und Radius des rotierenden Kreises

Tabelle 1: Übersicht der zur Zeit implementierten Flächentypen
(siehe `http://www.tu-chemnitz.de/~pester/facegeo.html`)

Ebenen werden normalerweise nicht als *spezielle* Geometrie behandelt, da die neuen Gitterpunkte bei linearer Netzverfeinerung automatisch in derselben Ebene bleiben. Eine zu beachtende Ausnahme stellen die Punkte an Schnittkanten zwischen einer Ebene



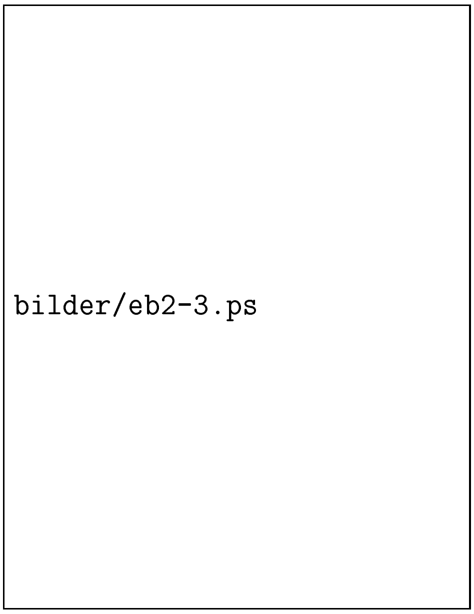


Abbildung 3: Vergleich zweier Netze, die Schnittebene wurde nur im rechten Bild explizit deklariert; stärkere Linien markieren das Grobnetz

und einer anderen Fläche dar. Nur solche Ebenen müssen explizit in der Geometriedefinition des Eingabe-Files auftreten. Diese Notwendigkeit wird aus dem Vergleich der beiden Netze in Abb. 3 deutlich.

3 Geometriebezogene Netzverfeinerung

3.1 Prinzipielles Vorgehen

Die Betrachtung von Faces mit speziellen geometrischen Eigenschaften ist unabhängig von der Art der Netzverfeinerung (gleichmäßig oder adaptiv). Ausgangspunkt ist ein Grobnetz

$$\Omega_0 = \{V_0, E_0, F_0, S_0\}\,,$$

bestehend aus

einer Menge	$V_0 = \left\{v_k^{(0)}\right\}$	von Knoten,
einer Menge	$E_0 = \left\{e_k^{(0)}\right\}$	von Kanten, definiert durch je 2 Knoten,
einer Menge	$F_0 = \left\{f_k^{(0)}\right\}$	von Faces, definiert durch je 3 oder 4 Kanten, und
einer Menge	$S_0 = \left\{s_k^{(0)}\right\}$	von Volumenelementen, definiert durch 4 oder 6 Faces.

Durch die Netzverfeinerung entsteht eine Folge von Netzen $\Omega_i = \{V_i, E_i, F_i, S_i\}$ für $i = 1, 2, 3, \ldots$. Es gelten die bekannten hierarchischen Beziehungen zwischen Level i und Level $i+1$:

$$V_i \subset V_{i+1}, \quad e_k^{(i)} = \bigcup_{j=1}^{n_1} e_{k_j}^{(i+1)}, \quad f_k^{(i)} = \bigcup_{j=1}^{n_2} f_{k_j}^{(i+1)}, \quad s_k^{(i)} = \bigcup_{j=1}^{n_3} s_{k_j}^{(i+1)},$$

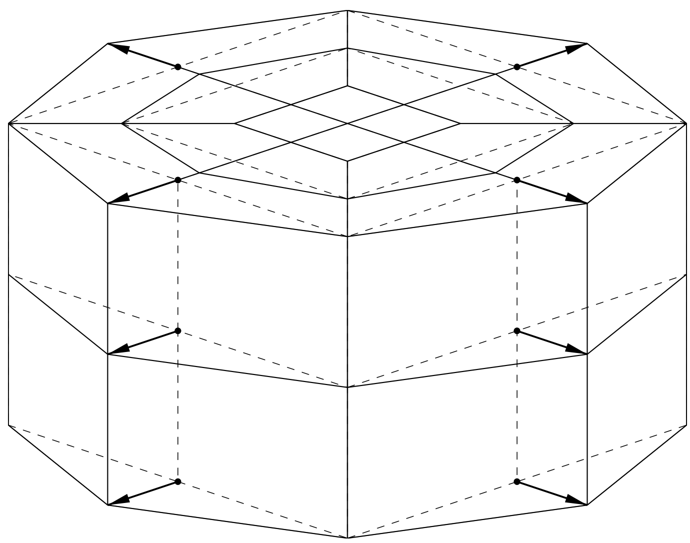


Abbildung 4: Projektion der neu entstehenden Knoten auf den Zylindermantel

wobei mit n_q die Anzahl der bei der Unterteilung eines q-dimensionalen Objektes (q-Face) entstehenden q-dimensionalen Teilobjekte ist ($n_1 = 2, n_2 = 4, n_3 = 8$ bei gleichmäßiger Unterteilung von Tetraedern oder Hexaedern).

Darüber hinaus existiert eine Menge $G = \{g_j\}$ aller für den Körper speziell zu definierenden Flächengeometrien. Während der Netzverfeinerung bleibt diese Menge unverändert erhalten. Einzelne $f_k^{(0)}$ sind markiert, indem sie ein *Flächenattribut* g_j tragen, welches sich beim Verfeinern auf die Tochter-Faces $f_{k_j}^{(i+1)}$ vererbt. Das Flächenattribut g_j muß offensichtlich bei allen neu entstehenden Knoten berücksichtigt werden, die aus dem Face $f_k^{(i)}$ oder den sie begrenzenden Kanten hervorgehen. Solche neuen Knoten $v_{k_j}^{(i+1)}$ werden zunächst durch lineare Verfeinerung (Mittelung an einer Kante oder bei Hexaeder-Elementen auch Face- und Volumen-Mittelpunkte) angenähert und anschließend auf die durch g_j definierte Fläche projiziert (Abb. 4). Die Art und Weise dieser Projektion wird im nachfolgenden Abschnitt an Beispielen erläutert.

Kanten werden aber stets von mehreren Faces begrenzt, so daß hier auch verschiedene Flächengeometrien zusammentreffen können. Durch Beschränkung auf maximal zwei verschiedene (spezielle) Geometrien an einer Kante ist das Problem auf die Betrachtung der Schnittkurve zweier Flächen zu reduzieren. Dies ist auch ausreichend, wenn nur an der Oberfläche eines Körpers gkrümmte Flächen auftreten. Der Fall des Zusammentreffens von mehr als zwei gekrümmten Flächen in genau **einer** Schnittkante kann als nahezu unrealistisch betrachtet werden und soll hier keine Rolle spielen.

Jede Kante besitzt folglich bis zu zwei verschiedene Flächenattribute. In diesem Fall ist gegenüber der einfachen Projektion auf eine Fläche eine Sonderbehandlung zur Projektion auf die Schnittkurve erforderlich.

Damit kann die Netzverfeinerung auf einem Parallelrechner nach der Verteilung der Elemente eines relativ groben Netzes auf die Prozessoren, einschließlich der Liste der Geometrie-Daten, ohne weitere Kommunikation völlig lokal ausgeführt werden.

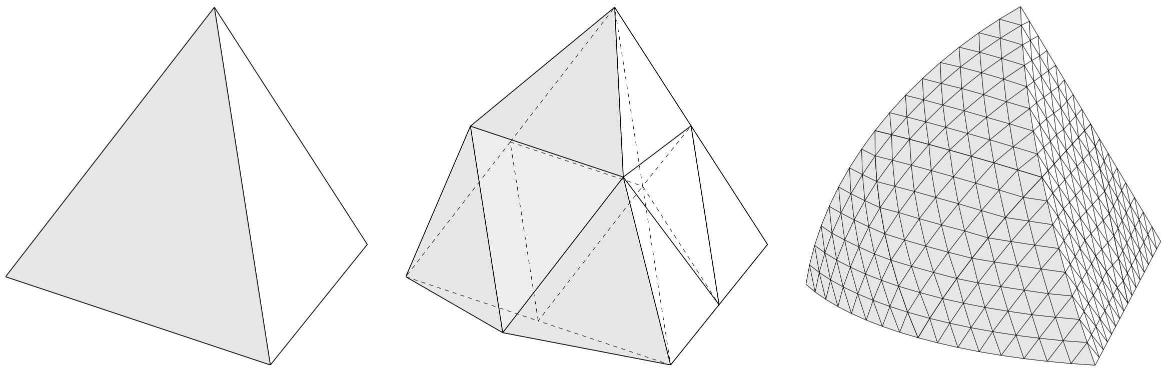


Abbildung 5: Vererbung der Flächengeometrie bei einem Tetraeder

3.2 Tetraedernetze

Tetraedernetze sind bezüglich ihrer Topologie relativ einfach zu behandeln. Hinsichtlich der Vererbung geometrischer Eigenschaften sind folgende Fälle zu beachten (Abb. 5):

- Neue Knoten entstehen grundsätzlich durch Unterteilung von Kanten. Eine von der linearen Unterteilung der Kante abweichende Korrektur der Knotenkoordinaten ist erforderlich, wenn für die Kante eine Markierung mit mindestens einer Flächengeometrie g_j besteht, also mindestens eines der angrenzenden Dreiecke zu einer durch g_j beschriebenen Fläche gehört.
- Die Tochter-Faces erben die Geometrie des Faces, von dem sie abstammen.
- Innerhalb eines Faces neu entstehende Kanten erhalten dadurch dieselbe Markierung (und nur diese), die das Face vorher hatte.
- Die im Inneren eines Tetraeders generierten Kanten und Faces werden ohne spezielle Geometrie-Attribute weiterbehandelt. Anderenfalls müßte eine Transformationsvorschrift der aktuellen Geometrie auf die inneren Flächen unter Beachtung einer gewissen Umgebung des Tetraeders angegeben werden können.

Pentaeder besitzen bezüglich der Vererbung der Geometrie von Faces und Kanten dieselben Eigenschaften wie Tetraeder.

3.3 Hexaedernetze

Gegenüber einer Tetraedervernetzung erfordert die Verfeinerung eines Hexaedernetzes mehr Beachtung, da hier nicht nur Kanten-, sondern auch Flächen- und Volumenmittelpunkte als neue Knoten zu bestimmen sind.
Bei nur geraden Kanten ist die lineare Unterteilung eines Hexaeders ausreichend:

- Berechnung der Kantenmitten aus den beiden Eckpunkten;
- Berechnung der Flächenmitten aus den vier Eckpunkten oder den vier Kantenmitten (bei ebenen Flächen sind beide Ergebnisse identisch);

- Berechnung des Volumenmittelpunktes aus den 8 Eckpunkten oder den 6 Flächenmitten.

Sobald aber mindestens eine Hexaederseite zu einer gekrümmten Fläche gehört, liefert diese Mittelung allein keine akzeptablen Ergebnisse. Betrachten wir zunächst die Flächenmittelpunkte eines zu teilenden Hexaeders.

Handelt es sich um den Mittelpunkt einer Fläche, die mit einer speziellen Geometrie versehen ist, so stellt die Projektion des neuen Punktes auf diese Fläche kein zusätzliches Problem gegenüber der Behandlung von Kantenmittelpunkten dar. Kritischer ist der Fall, daß eine oder mehrere Begrenzungskanten einer Hexaederfläche durch den Einfluß der angrenzenden Fläche krummlinig sind. Da in diesem Fall ein arithmetisches Mittel, auch unter Einbeziehung der bereits verschobenen Mittelpunkte der gekrümmten Kanten, keine brauchbaren Ergebnisse liefert (Abb. 6), müssen die Punkte im Inneren eines Faces als Linearkombination bestimmt werden

$$x^{(m)} = \sum_{i=1}^{8} x_i \varphi_i^{(8)} \left(\xi^{(m)}\right),$$

wobei die $\varphi_i^{(8)}$ die quadratischen Formfunktionen eines 8-Knoten-Serendipity-Elementes auf dem Referenzelement $[0,1] \times [0,1]$ sind (vgl. [15], [5]) und $\xi^{(m)} = \left(\frac{1}{2}, \frac{1}{2}\right)$ der Mittelpunkt dieses Referenzelementes ist. Der Ansatz liefert die Berechnung des Face-Mittelpunktes aus den 4 Ecken $x_k^{(v)}$ und 4 Kantenmittelpunkten $x_k^{(e)}$ der Hexaederfläche:

$$x^{(m)} = \frac{1}{2} \sum_{k=1}^{4} x_k^{(e)} - \frac{1}{4} \sum_{k=1}^{4} x_k^{(v)}$$

Derselbe Effekt entsteht bei der Berechnung des Mittelpunktes eines Hexaeder-Elementes, dessen Flächen gekrümmt sind (Abb. 8). Eine geometriebezogene Mittelung ist auch hier durch die Linearkombination

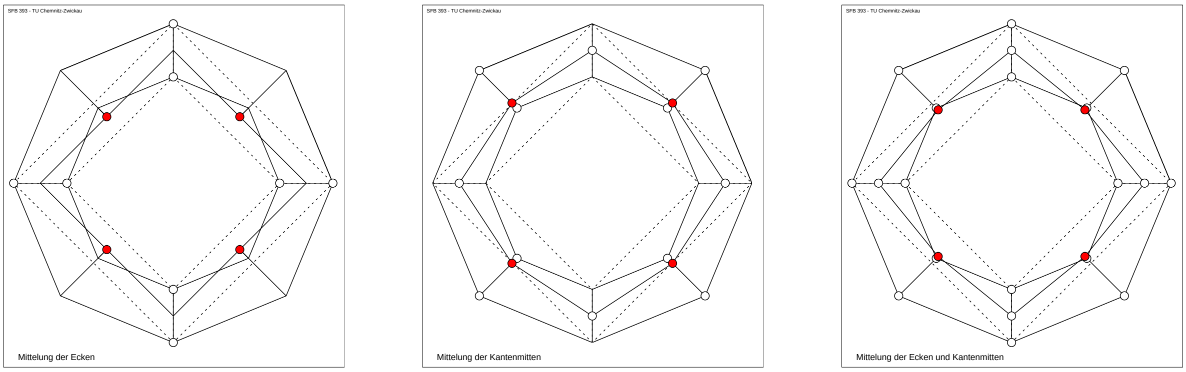
$$x^{(m)} = \sum_{i=1}^{20} x_i \varphi_i^{(20)} \left(\xi^{(m)}\right)$$

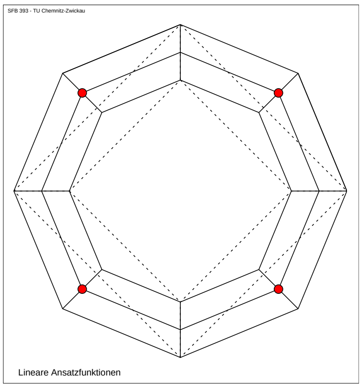
aus den entsprechenden Formfunktionen $\varphi_i^{(20)}$ des 20-Knoten-Serendipity-Elementes auf dem Referenzelement $[0,1] \times [0,1] \times [0,1]$ zu bestimmen.

Die Auswertung im Mittelpunkt $\xi^{(m)} = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ führt auf die folgende Berechnungsvorschrift aus den 8 Ecken $x_k^{(v)}$ und 12 Kantenmittelpunkten $x_k^{(e)}$:

$$x^{(m)} = \frac{1}{4} \sum_{k=1}^{12} x_k^{(e)} - \frac{1}{4} \sum_{k=1}^{8} x_k^{(v)}.$$

Durch diese Berechnung wird erreicht, daß der neue Punkt in jeder Raumrichtung in der Mitte des Hexaeder-Elementes bezüglich aller seiner Randknoten liegt und nicht wie in Abb. 8 (links) zu einer Seite verschoben ist, hier in Richtung des gemeinsamen Krümmungsmittelpunktes der Innen- und Außenflächen.





oben: arithmetische Mittelung aus den Koordinaten
der Eckpunkte,
der Kantenmitten,
der Ecken und Kantenmitten

links: Verwendung von Formfunktionen über Ecken und Kantenmitten

Gestrichelte Linien markieren die vier zu unterteilenden Hexaeder-Elemente

Abbildung 6: Verschiedene Resultate der Berechnung der Face-Mittelknoten bei Hexaeder-Elementen am Beispiel eines zylindrischen Rohrstücks

`bilder/rohr3a.ps` `bilder/rohr3b.ps`

Abbildung 7: Auswirkung der unterschiedlichen Berechnung der Face-Mittelknoten nach dreimaliger Verfeinerung eines aus 8 Hexaeder-Elementen bestehenden Grobnetzes; (links: bei arithmetischer Mittelung)

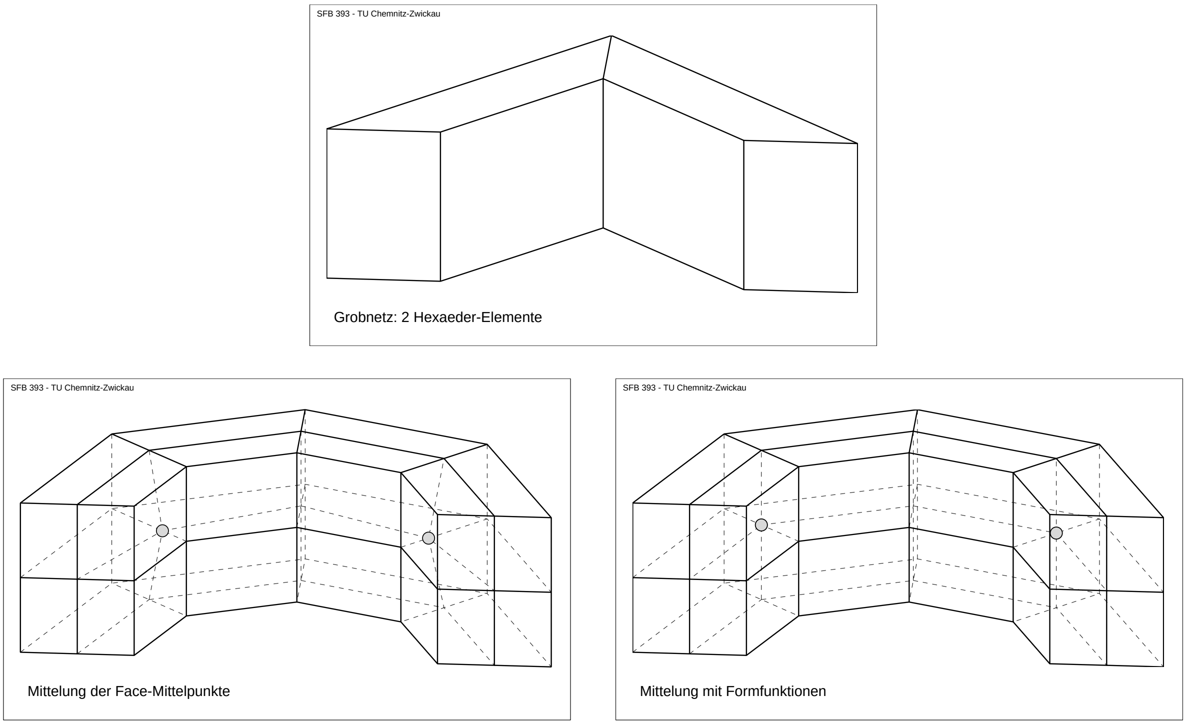


Abbildung 8: Mittelpunkt von Hexaeder-Elementen mit gekrümmten Flächen;
oben: zwei Hexaeder-Elmente sind zu unterteilen
links: verschobener Mittelpunkt bei arithmetischer Mittelung
rechts: korrekte Bestimmung über Formfunktionen

4 Projektionen auf gekrümmte Flächen

Nachfolgend werden einige Beispiele kurzer Algorithmen aufgeführt, mit denen nach einer linearen Unterteilung einer Kante (oder eines Faces) der berechnete Näherungspunkt $\tilde{x}$ auf die gekrümmte Fläche projiziert wird, in der die Kante (das Face) laut Geometriedefinition des Ausgangsnetzes liegen soll. Eine solche Implementierung erweist sich als relativ einfach, indem die Koordinaten des gesuchten neuen Punktes x^* als Funktion der berechneten Näherung und der aktuellen Flächengeometrie g bestimmt werden:

$$x^* = F\left(\tilde{x}, g\right).$$

Bei Schnittkanten zweier verschiedener Flächengeometrien g_1, g_2 sind beide Geometrie-Angaben zu berücksichtigen:

$$x^* = F\left(\tilde{x}, g_1, g_2\right).$$

4.1 Projektion auf eine Fläche

4.1.1 Zylindermantel

Der Zylinder ist nach Tabelle 1 (S. 4) definiert durch die beiden Achsenpunkte p_1, p_2 und den Radius r, d. h. $g_{11} = \{p_1, p_2, r\}$. Die Transformation F ist in diesem Fall eine radiale Verschiebung des Punktes $\tilde{x}$ in den Punkt x^* (Abb. 9).

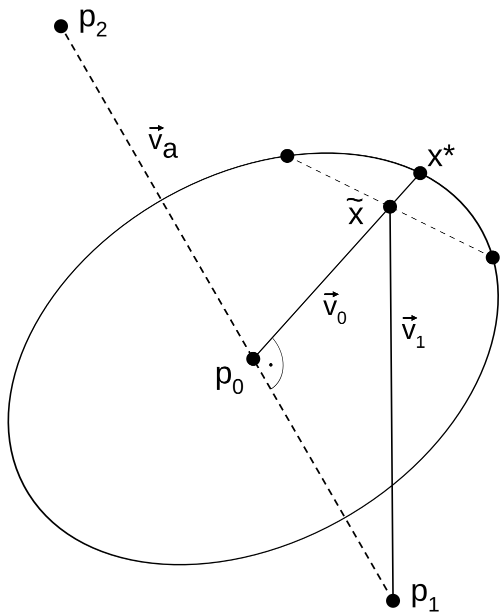


Abbildung 9: Projektion des Punktes $\tilde{x}$ auf den Zylindermantel

Der Abstand des Punktes $\tilde{x}$ von der Zylinderachse muß in Richtung des Vektors $\vec{v} = \tilde{x} - p_0$ auf den Wert r vergrößert werden. Wir bezeichnen

$\vec{v}_a$	$=$	$p_2 - p_1$	Richtung der Achse
$\vec{v}_0$	$=$	$\tilde{x} - p_0$	Vektor von der Achse zu $\tilde{x}$, $\quad \vec{v}_0 \perp \vec{v}_a$
$\vec{v}_1$	$=$	$\tilde{x} - p_1$	Vektor von einem gegebenen Achsenpunkt zu $\tilde{x}$

Der Punkt p_0 (der Punkt auf der Zylinderachse mit dem geringsten Abstand zu $\tilde{x}$) muß dazu nicht explizit bestimmt werden. Die Verschiebung wird durch elementare Vektoroperationen ausgeführt. Aus den gegebenen Größen $p_1, p_2, r, \tilde{x}$ wird x^* nach folgendem Algorithmus berechnet:

$$
\begin{aligned}
\vec{v}_a &:= p_2 - p_1 \\
\vec{v}_1 &:= \tilde{x} - p_1 \\
\alpha &:= (\vec{v}_a \cdot \vec{v}_a) \\
\beta &:= (\vec{v}_a \cdot \vec{v}_1) \\
\vec{v}_0 &:= \vec{v}_1 - \frac{|\vec{v}_1| \cos \sphericalangle(\vec{v}_a, \vec{v}_1)}{|\vec{v}_a|} \vec{v}_a \;=\; \vec{v}_1 - \frac{\beta}{\alpha} \vec{v}_a \\
x^* &:= \tilde{x} + (r - |\vec{v}_0|) \frac{\vec{v}_0}{|\vec{v}_0|} \;=\; \tilde{x} + \left(\frac{r}{|\vec{v}_0|} - 1 \right) \vec{v}_0
\end{aligned}
$$

Mit drei Skalarprodukten $(\alpha, \beta, |\vec{v}_0|)$ und vier Vektoradditionen (`saxpy`) für Koordinatenvektoren hält sich der Berechnungsaufwand in Grenzen.

Abbildung 10 zeigt, wie aus einem groben Startnetz durch Netzverfeinerung ein Zylinder entsteht.

4.1.2 Kugeloberfläche

Die Kugeloberfläche ist definiert durch $g_{21} = \{p_m, r\}$. Die Transformation von $\tilde{x}$ auf x^* ist hier einfach die Verschiebung radial vom Mittelpunkt p_m in Richtung $\vec{v}_0 = \tilde{x} - p_m$ bis zum Abstand r. Der Algorithmus ist entsprechend kurz:

$$\begin{aligned} \vec{v}_0 &:= \tilde{x} - p_m \\ x^* &:= p_m + r\frac{\vec{v}_0}{|\vec{v}_0|} \end{aligned}$$

Beispiele sind in den Bildern 11 und 12 dargestellt, in letzterem ist eine spezielle Deklaration der drei Schnittebenen nicht erforderlich, da sie alle den Kugelmittelpunkt enthalten und damit jede Projektion von Punkten der Außenkante in radialer Richtung in der Ebene bleibt.

4.1.3 Kegelmantel

Der Kegel ist definiert durch $g_{31} = \{p_1, p_2, r_2\}$. Bei Verschiebung des Näherungspunktes $\tilde{x}$ orthogonal zur Rotationsachse des Kegels ergibt sich die gleiche Aufgabe wie beim Zylinder, wobei hier der Radius proportional zum Abstand von p_1 bestimmt wird (Abb. 13). Auch hier muß der Punkt p_0 nicht explizit bestimmt werden.

Der leicht modifizierte Algorithmus sieht dann folgendermaßen aus:

$$\begin{aligned} \vec{v}_a &:= p_2 - p_1 \\ \vec{v}_1 &:= \tilde{x} - p_1 \\ \alpha &:= (\vec{v}_a \cdot \vec{v}_a) \\ \beta &:= (\vec{v}_a \cdot \vec{v}_1) \\ \vec{v}_0 &:= \vec{v}_1 - \frac{\beta}{\alpha}\vec{v}_a \\ r_0 &:= \frac{|p_0 - p_1|}{|\vec{v}_a|}r_2 = \frac{\beta}{\alpha}r_2 \\ x^* &:= \tilde{x} + (r_0 - |\vec{v}_0|)\,\frac{\vec{v}_0}{|\vec{v}_0|} \;=\; \tilde{x} + \left(\frac{r_0}{|\vec{v}_0|} - 1\right)\vec{v}_0 \end{aligned}$$

4.1.4 Oberfläche von Rotationskörpern

Wir betrachten hier Oberflächen von Rotationshyperboloiden oder Rotationsellipsoiden, deren Geometrie definiert ist durch die Parameter $g_{41} = \{p_1, p_2, A, B\}$. Dabei sei p_1 Koordinatenursprung eines lokalen (ξ, η)-Koordinatensystems, dessen durch die Richtung $\overrightarrow{p_1 p_2}$ definierte η-Achse die Rotationsachse ist. Die ξ-Achse ist die *„rotierende“* Achse. Die aktuelle (ξ, η)-Ebene ist durch die (feste) η-Achse und den Näherungspunkt $\tilde{x}$ bestimmt. Die rotierende Kurve ist definiert durch

$$\pm\frac{\xi^2}{a^2} \pm \frac{\eta^2}{b^2} = 1, \qquad \text{mit } A = \pm a^2, B = \pm b^2.$$

Für $A > 0, B > 0$ entsteht ein Rotationsellipsoid, bei $A > 0, B < 0$ ein einschaliges Hyperboloid (Abb. 15, links) und bei $A < 0, B > 0$ ein zweischaliges Hyperboloid `bilder/zyl5-lv0.ps` `bilder/zyl5-lv3.ps`

Abbildung 10: Datenfile `mesh4/zyl5.std`: 5 Hexaeder-Elemente, mit Zylindergeometrie an den 4 Außenflächen

`bilder/kug20-0.ps` `bilder/kug20-3.ps`

Abbildung 11: Datenfile `mesh3/kugel20.std`: 20 Tetraeder-Elemente (Ikosaeder), mit Kugelgeometrie an allen Außenflächen

`bilder/fi-kug0.ps` `bilder/fi-kug4.ps`

Abbildung 12: Datenfile `mesh3/fi-kug.std`: 7 Tetraeder-Elemente, Kugel mit ausgeschnittenem Sektor

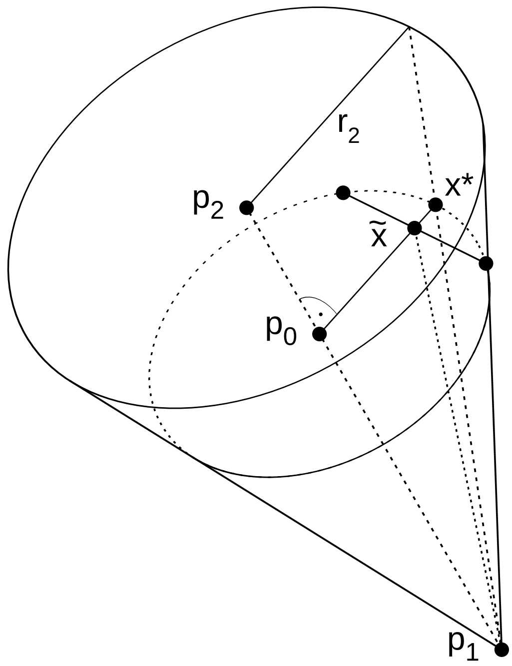


Abbildung 13: Projektion des Punktes $\tilde{x}$ auf den Kegelmantel

`bilder/k4n-0.ps`

`bilder/k4n-4.ps`

Abbildung 14: Datenfile `mesh3/kegel4.std`: 4 Tetraeder-Elemente, mit Kegelgeometrie an den vier Seitenflächen

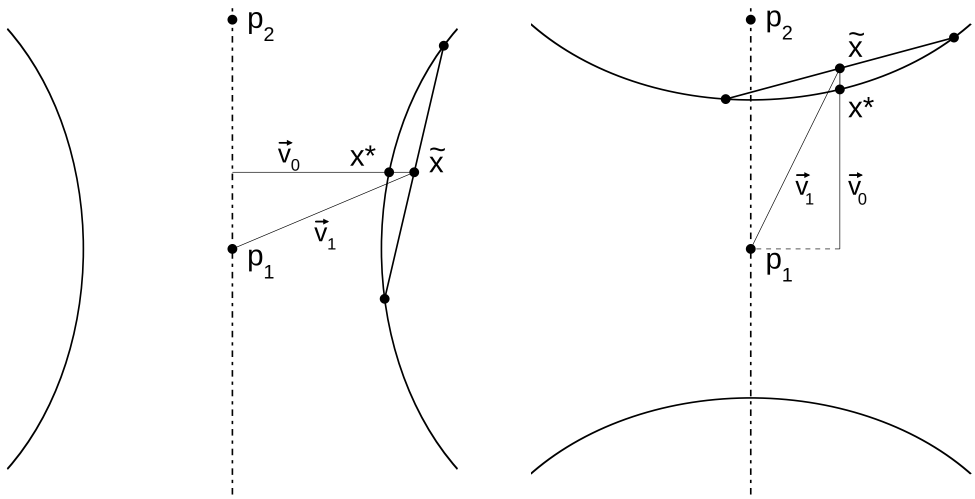


Abbildung 15: Projektion des Punktes $\tilde{x}$ auf ein einschaliges bzw. zweischaliges Rotationshyperboloid

(Abb. 15, rechts). Die Korrektur von $\tilde{x}$ zu x^* soll durch Verschiebung senkrecht zur Rotationsachse erfolgen, bzw. parallel zu dieser Achse im letztgenannten Fall. Daraus ergibt sich folgende Berechnung:

$$\begin{aligned}
\vec{v}_a &:= p_2 - p_1 \\
\vec{v}_1 &:= \tilde{x} - p_1 \\
\alpha &:= (\vec{v}_a \cdot \vec{v}_a) \\
\beta &:= (\vec{v}_a \cdot \vec{v}_1)
\end{aligned}$$

im Fall $A > 0$: ($\vec{v}_0 \perp \vec{v}_a$)

$$\begin{aligned}
\vec{v}_0 &:= \vec{v}_1 - \frac{\beta}{\alpha}\vec{v}_a \\
\eta^2 &:= \frac{\beta^2}{\alpha} \\
r &:= \sqrt{A\left(1 - \frac{\eta^2}{B}\right)}
\end{aligned}$$

im Fall $A < 0$: ($\vec{v}_0 \parallel \vec{v}_a$)

$$\begin{aligned}
\vec{v}_0 &:= \frac{\beta}{\alpha}\vec{v}_a \\
\xi^2 &:= (\vec{v}_1 \cdot \vec{v}_1) - \frac{\beta^2}{\alpha} \\
r &:= \sqrt{B\left(1 - \frac{\xi^2}{A}\right)}
\end{aligned}$$

$$x^* := \tilde{x} + \left(\frac{r}{|\vec{v}_0|} - 1\right)\vec{v}_0$$

Beispiele sind in den Abbildungen 16 und 17 dargestellt. Es soll noch vermerkt sein, daß die gewählte Vorgehensweise mit einer Projektion senkrecht oder parallel zur Rotationsachse offenbar gute Ergebnisse liefern, wenn die betrachteten Punkte in der Nähe der Scheitelpunkte der Hyperbel liegen. Andererseits ist bei Annäherung einer Hyperbel an ihre Asymptoten bereits die lineare Unterteilung eine gute Näherung für den exakten Punkt. Die beschriebene Projektion führt zu keiner nennenswerten Deformation des Netzes.

bilder/hyp-0.ps

bilder/hyp-4.ps

Abbildung 16: Datenfile `mesh4/hyp.std`: Aus 5 Hexaeder-Elementen wird ein Hyperboloid

bilder/hyp2-0.ps

bilder/hyp2-3.ps

Abbildung 17: Datenfile `mesh3/hyp2.std`: Zwei gegenüberliegende Flächen sind hier als zweischaliges Hyperboloid definiert.

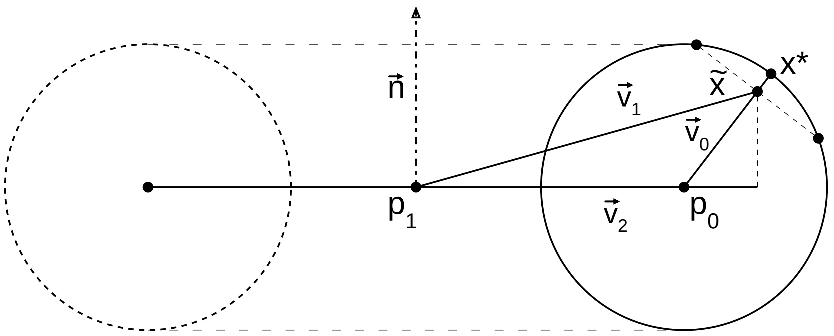


Abbildung 18: Projektion des Punktes $\tilde{x}$ auf einen Torus

4.1.5 Torusfläche

Die Geometrie der Torusfläche wird beschrieben durch $g_{51} = (p_1, \vec{n}, r_1, r_2)$. Der Torus entsteht durch Rotation eines Kreises mit dem Radius r_2 um eine Achse, wobei der Kreismittelpunkt den Abstand r_1 von dieser Achse hat. Die Richtung der Achse ist durch $\vec{n}$ gegeben und p_1 ist der in der Rotationsebene liegende Punkt der Achse. Die Projektion des Näherungspunktes $\tilde{x}$ auf den Punkt x^* erfolgt radial bezüglich des rotierenden Kreises. Durch $p_1, \vec{n}, \tilde{x}$ wird eine Ebene bestimmt, in der auch der Mittelpunkt p_0 dieses Kreises liegt. Dann liegt x^* im Abstand r_2 von p_0 in Richtung $\overrightarrow{p_0\tilde{x}}$ (Abb. 18) und läßt sich aus den gegebenen Größen folgendermaßen berechnen:

$$
\begin{aligned}
\vec{v}_1 &:= \tilde{x} - p_1 \\
\alpha &:= (\vec{n} \cdot \vec{n}) \\
\beta &:= (\vec{n} \cdot \vec{v}_1) \\
\vec{v}_2 &:= \vec{v}_1 - \frac{\beta}{\alpha}\vec{n} \\
\vec{v}_0 &:= \vec{v}_1 - \frac{r_1}{|\vec{v}_2|}\vec{v}_2 \qquad (= \tilde{x} - p_0) \\
x^* &:= \tilde{x} + (r_2 - |\vec{v}_0|)\frac{\vec{v}_0}{|\vec{v}_0|} \;=\; \tilde{x} + \left(\frac{r_2}{|\vec{v}_0|} - 1\right)\vec{v}_0
\end{aligned}
$$

Der Torus in Abb. 21 zeigt eine leichte Verschiebung der Knoten auf der Oberfläche, die durch die Projektion in Richtung des Mittelpunktes entsteht, wenn $\tilde{x}$ aus der Verfeinerung einer zur Grundebene parallelen Kante hervorgeht. Dieser Effekt ist unvermeidlich, wenn nicht in die Berechnung von x^* neben $\tilde{x}$ auch die ursprüngliche Lage der Punkte eingeht, aus denen $\tilde{x}$ berechnet wurde.

4.2 Projektion auf Schnittkanten

4.2.1 Zylinder und Ebene

Wird ein Zylinder von einer Ebene nicht orthogonal[1] zur Achse geschnitten, dann soll die Projektion von $\tilde{x}$ auf den Zylindermantel nicht mehr radial, senkrecht von der Achse

[1]Natürlich darf eine Schnittebene senkrecht zur Zylinderachse explizit definiert sein ($\vec{n} \| \vec{v}_a$), aber dasselbe Ergebnis entsteht auch bei alleiniger Betrachtung der Zylindergeometrie.

bilder/torus-0.ps

bilder/torus-3.ps

Abbildung 19: Datenfile `mesh3/torus48.std`: 48 Tetraeder-Elemente sind zum Torus verbunden

bilder/4tltor-0.ps

bilder/4tltor-3.ps

Abbildung 20: Datenfile `mesh3/4tltor8.std`: 96 Tetraeder-Elemente, Viertel eines Torus, feineres Startnetz und andere Radien als Bild 19

bilder/tor4-0.ps

bilder/zylring3.ps

bilder/tor4-3.ps

Abbildung 21: Datenfiles `mesh4/torus40.std, mesh4/ring40.std`:
40 Hexaeder-Elemente; durch unterschiedliche Flächengeometrie wird daraus ein Ring aus 8 Zylinderstücken oder ein Torus.

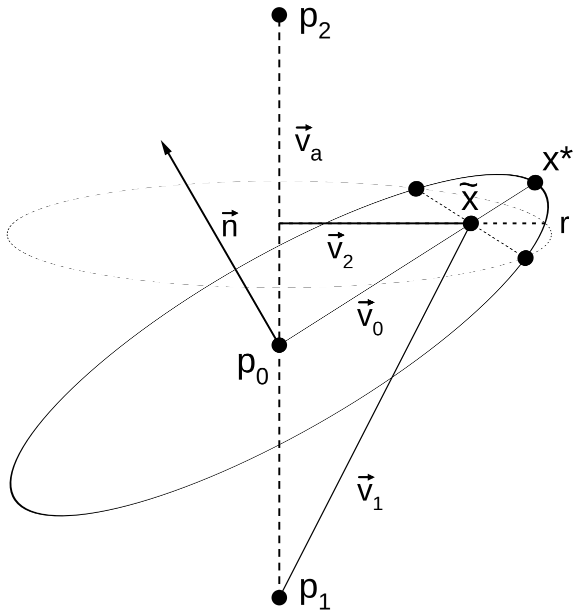


Abbildung 22: Projektion des Punktes $\tilde{x}$ auf eine Schnittkante zwischen Ebene und Zylinder

nach außen erfolgen, sondern innerhalb der Schnittebene (Abb. 22). Gegebene Parameter sind die Geometrien $g_{11} = \{p_1, p_2, r\}$, $g_{01} = \{\vec{n}, p_0\}$. Der Punkt p_0 ist für die Berechnungen hier ohne Bedeutung, da man davon ausgehen kann, daß die zu unterteilende Kante vollständig zur Ebene gehört und somit auch der Punkt $\tilde{x}$. Unabhängig von der Definition in g_{01} bezeichnen wir mit p_0 den Schnittpunkt der Ebene mit der Zylinderachse. Im Unterschied zur einfachen Projektion auf den Zylindermantel (4.1.1) gilt nun **nicht** mehr $\vec{v}_0 \perp \vec{v}_a$, sondern $\vec{v}_0 \perp \vec{n}$. Der neue Punkt $x^* = F(\tilde{x}, g_{01}, g_{21})$ auf der Schnittkante berechnet sich auf folgende Weise:

$$
\begin{aligned}
\vec{v}_a &:= p_2 - p_1 \\
\text{Falls}\,(\vec{v}_a \cdot \vec{n}) &= 0, \text{ ist die Ebene parallel zum Zylindermantel} \\
& \quad\ \text{und folglich } x^* = \tilde{x}, \\
\text{anderenfalls:} & \\
\vec{v}_1 &:= \tilde{x} - p_1 \\
\alpha_a &:= (\vec{v}_a \cdot \vec{v}_a) \\
\alpha_n &:= (\vec{n} \cdot \vec{n}) \\
\beta_a &:= (\vec{v}_1 \cdot \vec{v}_a) \\
\beta_n &:= (\vec{v}_1 \cdot \vec{n}) \\
\vec{v}_2 &:= \vec{v}_1 - \frac{\beta_a}{\alpha_a}\vec{v}_a \\
\vec{v}_0 &:= \vec{v}_1 - \frac{\beta_n}{\alpha_n}\vec{n} \\
x^* &:= \tilde{x} + \frac{(r - |\vec{v}_2|)}{|\vec{v}_2|}\frac{\vec{v}_0}{|\vec{v}_0|} = \tilde{x} + \left(\frac{r}{|\vec{v}_2|} - 1\right)\frac{\vec{v}_0}{|\vec{v}_0|}
\end{aligned}
$$

Alternativ könnte in Betracht gezogen werden, den Punkt x^* auf der Schnittkurve so zu bestimmen, daß der Abstand $|x^* - \tilde{x}|$ minimal wird. Es ergibt sich jedoch eine relativ komplizierte Berechnung, aber keine Verbesserung der Approximation der Schnittkurve ([8]).

4.2.2 Kugel und Ebene

Bei einem ebenen Schnitt durch eine Kugel erfolgt die Korrektur des Punktes $\tilde{x}$ an einer Schnittkante innerhalb der Schnittebene in radialer Richtung, betrachtet von der Projektion M_0 des Kugelmittelpunktes M auf die Ebene, bis zur Kugeloberfläche (Abb. 23). Mit den gegebenen Geometrieparametern $g_{01} = \{\vec{n}, p_0\}$ und $g_{21} = \{M, r\}$ ergibt sich folgende Berechnung für $x^* = F(\tilde{x}, g_{01}, g_{21})$, wobei mit M_0, r_0 Mittelpunkt und Radius des Schnittkreises bezeichnet sind:

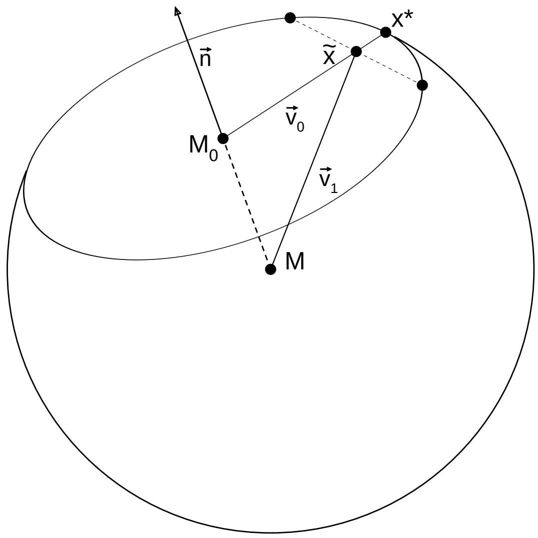


Abbildung 23: Projektion des Punktes $\tilde{x}$ auf eine Schnittkante zwischen Ebene und Kugel

`bilder/platkug0.ps` `bilder/platkug3.ps`

Abbildung 24: Datenfile `mesh3/platkug.std`: 20 Tetraeder-Elemente, Kugel mit ebener Schnittfläche.

$$\begin{aligned}
\vec{v}_1 &:= \tilde{x} - M \\
\alpha &:= (\vec{n} \cdot \vec{n}) \\
\beta &:= (\vec{v}_1 \cdot \vec{n}) \\
r_0 &:= \sqrt{r^2 - \left|\overline{MM}_0\right|^2} \;=\; \sqrt{r^2 - \frac{\beta^2}{\alpha}} \\
\vec{v}_0 &:= \vec{v}_1 - \left|\overline{MM}_0\right| \frac{\vec{n}}{|\vec{n}|} \;=\; \vec{v}_1 - \frac{\beta}{\alpha}\vec{n} \\
x^* &:= \tilde{x} + (r_0 - |\vec{v}_0|)\frac{\vec{v}_0}{|\vec{v}_0|} \;=\; \tilde{x} + \left(\frac{r_0}{|\vec{v}_0|} - 1\right)\vec{v}_0
\end{aligned}$$

Das Beispiel in Bild 24 hat im Grobnetz die gleiche Struktur wie die Kugel in Bild 11, wobei die oberen Punkte hier die Schnittebene bilden.

4.2.3 Kegel und Ebene

Nur wenig komplizierter als der ebene Schnitt eines Zylinders ist der Schnitt durch einen Kegel zu realisieren (Abb. 25). Als einfachste Lösung bietet sich an, den Punkt $\tilde{x}$ zunächst nach 4.1.3 senkrecht zur Kegelachse auf den Kegelmantel zu projizieren (x_h) und anschließend den Schnittpunkt x^* der von der Kegelspitze p_1 ausgehenden Geraden durch diesen Punkt mit der gegebenen Schnittebene zu bestimmen. Die Berechnung aus den gegebenen Größen $\tilde{x}$, $g_{31} = \{p_1, p_2, r_2\}$, $g_{01} = \{\vec{n}, p_0\}$ erfolgt dann so:

$$\begin{aligned}
\vec{v}_a &:= p_2 - p_1 \\
\vec{v}_1 &:= \tilde{x} - p_1 \\
\alpha &:= (\vec{v}_a \cdot \vec{v}_a) \\
\beta &:= (\vec{v}_a \cdot \vec{v}_1) \\
\vec{v}_0 &:= \vec{v}_1 - \frac{\beta}{\alpha}\vec{v}_a \\
r_0 &:= \frac{\beta}{\alpha} r_2 \\
x_h &:= \tilde{x} + \left(\frac{r_0}{|\vec{v}_0|} - 1\right)\vec{v}_0 \\
\vec{v}_2 &:= x_h - p_1 \;=\; \vec{v}_1 + \left(\frac{r_0}{|\vec{v}_0|} - 1\right)\vec{v}_0
\end{aligned}$$

falls $(\vec{v}_0 \cdot \vec{n}) = 0$, so ist $x^* = x_h$, sonst:

$$x^* \;:=\; p_1 + \frac{(\vec{v}_1 \cdot \vec{n})}{(\vec{v}_2 \cdot \vec{n})}\vec{v}_2$$

Der Hilfspunkt x_h braucht dabei offensichtlich nicht explizit berechnet zu werden, da nur der Vektor $\vec{v}_2$ verwendet wird.

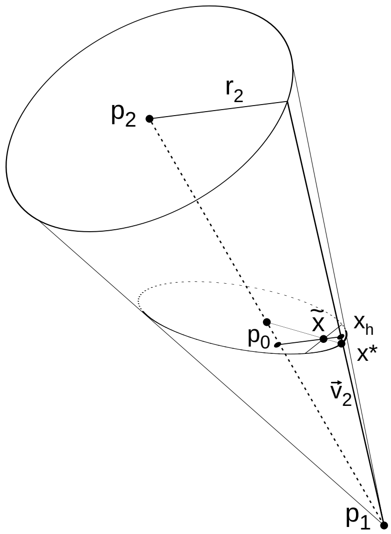


Abbildung 25: Projektion des Punktes $\tilde{x}$ auf die Schnittkante zwischen Ebene und Kegel

bilder/keg-eb0.ps

bilder/keg-eb4.ps

Abbildung 26: Datenfile `mesh3/kegel4e2.std`: 4 Tetraeder-Elemente, Kegel mit zwei ebenen Schnittflächen.

bilder/keg6s-0.ps

bilder/keg6s-4.ps

Abbildung 27: Datenfile `mesh3/kegel6s.std`: Kegelstumpf, begrenzt durch eine schiefe Ebene – **der** Kegelschnitt

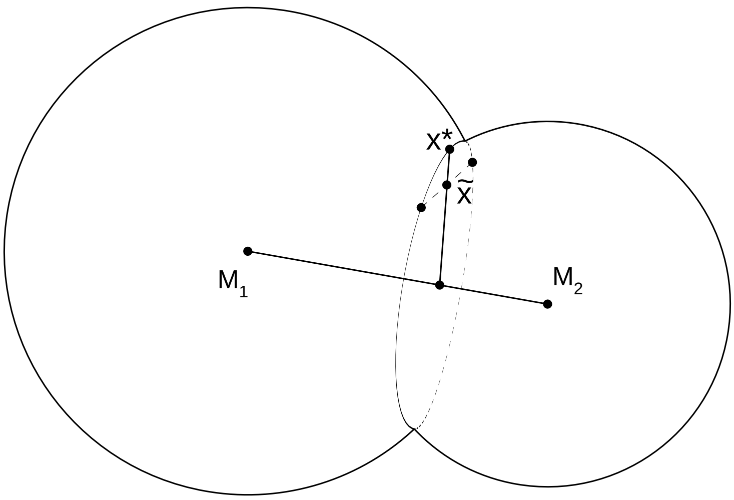


Abbildung 28: Schnittkanten bei zwei Kugeln

`bilder/glchdic0.ps`

`bilder/glchdick.ps`

Abbildung 29: Die Eckpunkte des Tetraeders sind die Mittelpunkte von vier sich schneidenden Kugeln

4.2.4 Zwei Kugeln

Als Schnittkante zweier Kugeln $g_{21}^{(1)} = \{M_1, r_1\}$, $g_{21}^{(2)} = \{M_2, r_2\}$ entsteht ein Kreis, der in einer Ebene senkrecht zur Verbindungsgeraden der beiden Kugelmittelpunkte liegt. Damit lassen sich solche Schnittkanten auf den soeben betrachteten Fall Kugel und Ebene zurückführen, indem eine der beiden Kugeln mit der durch den Normalenvektor $\overrightarrow{M_1 M_2}$ definierten Schnittebene betrachtet wird. Die Ebene ist dadurch eindeutig bestimmt, daß der Punkt $\tilde{x}$ bereits in ihr liegt.

Bild 29 zeigt ein Schnittgebilde aus vier Kugeln, deren Mittelpunkte die Ecken eines regulären Tetraeders sind.

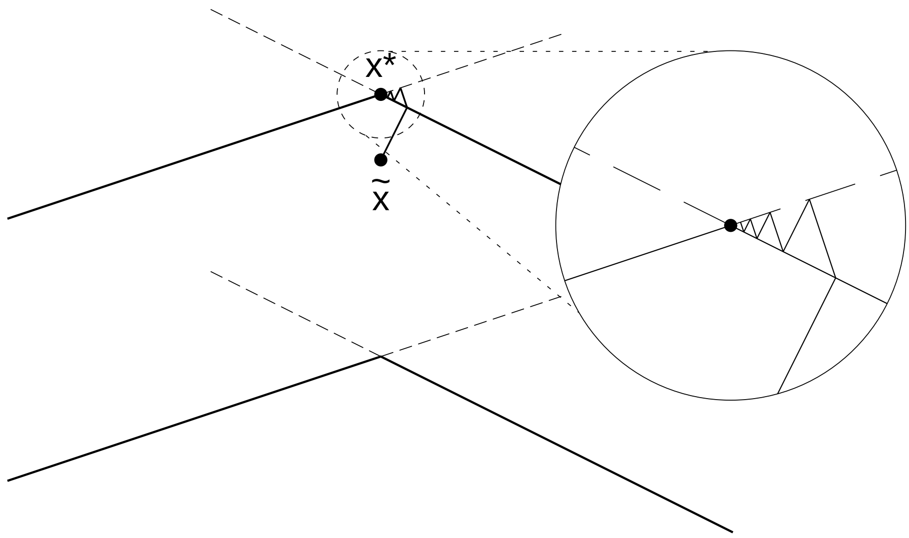


Abbildung 30: Iterative Berechnung von x^* auf einer Schnittkante zweier Zylinder

bilder/xzyl-0.ps

bilder/xzyl-3.ps

Abbildung 31: Durchdringung zweier Zylinder, Startnetz mit 96 Tetraedern

4.2.5 Zwei Zylinder und andere Schnittkanten

Ein Punkt x^* auf der Schnittkante zwischen zwei Zylindern $g_{11}^{(1)} = \{p_{11}, p_{12}, r_1\}$, $g_{11}^{(2)} = \{p_{21}, p_{22}, r_2\}$ wird aus $\tilde{x}$ durch eine kurze Iteration bestimmt, indem der Punkt alternierend (nach 4.1.1) auf die beiden Zylinder projiziert wird, bis er auf beiden Zylindern, also auf deren Schnittkante liegt. Falls die beiden Zylinderachsen orthogonal zueinander sind, führt diese Methode nach zwei Projektionen bereits zum korrekten Ergebnis. Bei beliebiger Lage zueinander können einige (wenige) Schritte mehr erforderlich sein, wie in Abb. 30 angedeutet.

Diese Methode kann allgemein für beliebige Schnittkanten zwischen Flächen unterschiedlicher Geometrie angewendet werden, sofern sich keine effektivere Speziallösung anbietet.

`bilder/zylq-0.ps` `bilder/zylq-3.ps`

Abbildung 32: Ungeeignete Definition eines Zylinders, da beim Verfeinern entartete Hexaeder-Elemente entstehen (vgl. Bild 10, S. 13)

4.3 Probleme bei der Definition der Geometrie im Grobnetz

Die vorgestellten Beispiele demonstrieren vor allem die Einfachheit der Algorithmen zur Verschiebung von Punkten auf eine Fläche mit vorgegebener Standard-Geometrie. Da die so verfeinerten FEM-Netze für numerische Berechnungen verwendet werden, sind zusätzliche Bedingungen zu erfüllen, insbesondere sollten keine entarteten Elemente entstehen. Das muß bereits bei der Definition des Ausgangsnetzes und der Flächengeometrien beachtet werden und stellt durchaus ein nichttriviales Problem dar (vgl. Abb. 32 und 33). Zum Beispiel wurde in Abb. 32 den Außenflächen eines Quaders eine Zylindergeometrie zugeordnet. Dabei entarten die Hexaeder-Elemente an den ursprünglichen Ecken des Ausgangselementes, da beim Verfeinern dort Innenwinkel von fast 180° entstehen. Dies ist vermeidbar, wenn der Quader, wie in Abb. 10 (S. 13) gezeigt, in 5 Hexaeder-Elemente zerlegt wird, so daß die Innenwinkel vor der Verfeinerung nur 45° statt 90° betragen, wobei eine Kante bzw. Fläche nicht auf dem Zylindermantel liegt.

Dasselbe Problem tritt auch bei Tetraeder-Elementen auf, wenn wie in Abb. 33 (hier die Grundfläche eines Kegels) im Grobnetz ein Innenwinkel von 90° an einer als *rund* spezifizierten Kante auftritt. Obwohl bei der Unterteilung der Tetraeder die von J. Bey [4] empfohlene Methode zur *stabilen Verfeinerung* verwendet wird, bei der im Normalfall nur sechs verschiedene Ähnlichkeitsklassen von Tetraedern entstehen, kommt es in diesem Fall zur Entartung von Tetraedern in der Nähe der gekrümmten Fläche. Auch hier genügt es, wenn im Grobnetz an solchen Ecken (Kanten) zusätzlich eine Kante (Fläche) ins Innere des Körpers zeigt, die für eine Begrenzung der Innenwinkel an gekrümmten Flächen sorgt.

Dagegen kann eine zylindrische Bohrung im Ausgangsnetz die Form eines Quaders haben, ohne daß dadurch ein solches Entartungsproblem auftritt (Abb. 35).

Es wird ebenfalls vorausgesetzt, daß die im Grobnetz definierten Gitterpunkte die entsprechenden Gleichungen der Flächengeometrien erfüllen, zu denen sie gehören (sollen). Bild 36 zeigt die Folgen einer fehlerhaften Geometriedefinition durch falsche Radius-Angabe für einen Zylinder. Alle durch Netzverfeinerung entstandenen Punkte liegen auf `bilder/pyunt-0.ps` `bilder/pyunt-4.ps`

`bilder/k4unt-0.ps` `bilder/k4unt-4.ps`

Abbildung 33: Auch entartete Tetraeder-Elemente entstehen, wenn die Geometrie im Grobnetz nicht ausreichend beachtet wird; hier verdeutlicht an der Ansicht der Grundfläche, oben: Kegel aus zwei Tetraedern, unten: Kegel aus vier Tetraedern

`bilder/kug7a.ps` `bilder/kug7a4.ps` `bilder/kug7c4.ps`

Abbildung 34: Ein Würfel, in 7 Hexaeder-Elemente zerlegt, kann auch zu einer akzeptablen Kugel verfeinert werden. Die mittlere Darstellung zeigt die Struktur des verfeinerten Netzes im Inneren.

`bilder/bohr-0.ps` `bilder/bohr-3.ps` `bilder/bohr-i.ps`

Abbildung 35: Eine Zylinderbohrung kann im Grobnetz quaderförmig sein, rechts die Isoliniendarstellung des Betrags einer elastischen Deformation

`bilder/zylfals3.ps`

Abbildung 36: Auswirkungen eines Fehlers in der Geometrie-Definition (Radius zu groß angegeben)

Abbildung 37: Drei Zwischenstufen zum Aufbau eines Netzes, das sich als Zylinder- oder Kugel definieren läßt

dem angegebenen Zylinder, die Punkte des Startnetzes aber nicht. Bei Bedarf könnte eine ähnliche Korrektur angewendet werden, wie sie hier für die bei der Netzverfeinerung entstehenden Gitterpunkte beschrieben wurde. Es gibt allerdings einen Unterschied. Hier wurde stets vorausgesetzt, daß der Näherungspunkt $\tilde{x}$ aus der linearen Unterteilung von Kanten entstand, die bereits auf der entsprechenden Fläche lagen. Insbesondere bei Ebenen wurde zur Vereinfachung angenommen, daß $\tilde{x}$ schon in der Ebene liegt. Gerade bei der Konstruktion komplexer Gebiete mit verschiedenen sich teilweise schneidenden Oberflächengeometrien ist eine entsprechende Sorgfalt erforderlich. Die Abbildungen machen deutlich, daß zur Konstruktion solcher Netze auch einige hilfreiche Werkzeuge benutzt werden sollten, wie etwa das von D. Lohse entwickelte Programm `geo_conv` für einfache geometrische Transformationen mit Zusammenfügen (*merge*) verschiedener Bauteile. Damit lassen sich solche komplexen Strukturen nach dem Baukastenprinzip aufbauen.

In Abb. 37 sind drei Etappen der schrittweisen Konstruktion eines aus 24 kongruenten Tetraedern zusammengesetzten Würfels angegeben (das Netz nach 3 Verfeinerungen soll lediglich den räumlichen Eindruck verdeutlichen). Im ersten Bild ist eine Doppelpyramide durch Spiegelung einer aus vier Tetraedern gebildeten Pyramide (vgl. Bild 14) an der x-y-Ebene entstanden. Dieselbe Doppelpyramide wurde im zweiten Bild nochmals nach einer 90°-Drehung um die x-Achse hinzugefügt und im dritten Bild nach einer Drehung um die y-Achse. Soll aus dem so entstandenen Würfel etwa eine Kugel oder ein Zylinder entstehen, so sind vorher die Flächenmittelpunkte auf den entsprechenden Wert (den $\sqrt{2}$- bzw. $\sqrt{3}$-fachen Abstand vom Mittelpunkt) zu korrigieren (Abb. 38, 39).

`bilder/cyl24-0.ps` `bilder/cyl24-3.ps`

Abbildung 38: Der Würfel aus Abb. 37 wird zum Zylinder ...

`bilder/kug24-0.ps` `bilder/kug24-3.ps`

Abbildung 39: ... oder zur Kugel umdefiniert `bilder/kapsel-0.ps`

`bilder/kapsel-4.ps`

Abbildung 40: Durch Hinzunahme weiterer (2×4) Tetraeder-Elemente wird aus dem Zylinder diese „Kapsel“.

4.4 Beispiele mit verschiedenen Geometrien

In den folgenden Abbildungen sind einige weitere Beispiele angegeben, die bei relativ grober Ausgangsvernetzung per Geometriedefinition zu Körpern mit gekrümmten Oberflächen *verfeinert* wurden.

`bilder/kamin0.ps` `bilder/kamin.ps`

Abbildung 41: Ein Kamin mit veränderlichem Querschnitt,
Startnetz: 25 Hexaeder-Elemente

bilder/zylbog-0.ps

bilder/zylbog-3.ps

bilder/zylbogn3.ps

Abbildung 42: Aus Zylinder- und Torus-Teilen zusammengesetzter Bogen, Startnetz: 504 Tetraeder-Elemente,
unten: Ergebnis einer elastischen Deformation

bilder/k3f-0.ps

bilder/k3f-3.ps

Abbildung 43: Eine Kurbelwelle, begrenzt von Ebenen und Zylinderflächen,
Startnetz: 123 Hexaeder-Elemente

bilder/tzyl8-0.ps

bilder/tzyl8-4.ps

Abbildung 44: Zylinder-T-Stück, Startnetz 72 Tetraeder-Elemente `bilder/z20dr-0.ps` `bilder/z20dr.ps`

Abbildung 45: Drei zylindrische, einander schneidende Bohrungen

5 Zusammenfassung

Es wurde eine Methode zur einfachen Beschreibung von geometrischen Oberflächeneigenschaften in einem FEM-Grobnetz betrachtet. Das Ziel war dabei, mit möglichst wenigen Parametern und grober Anfangsvernetzung auch kompliziertere Geometrien verarbeiten zu können. Damit kann die eigentliche Netzgenerierung, einschließlich der Erzeugung gekrümmter Flächen auf die Prozessoren eines Parallelrechners verteilt werden. Der Datentransfer vom Eingabefile und innerhalb des Parallelrechners bleibt somit minimal. Trotzdem ist das Grobnetz eine exakte Beschreibung der Geometrie, die je nach Gesamtspeicherkapazität im Parallelrechner beliebig genau durch ein FEM-Netz approximiert werden kann.

Die beschriebene Vorgehensweise ist besonders für eine konstruktive Beschreibung von Gebieten geeignet. Sie ist offen für Erweiterungen. Zu jeder Flächengeometrie, die sich mit einer gewissen Anzahl von Parametern beschreiben läßt, wird ein zusätzliches Unterprogramm zur Realisierung der Funktion $x^* := F(\tilde{x}, g)$ benötigt, um einen Näherungspunkt auf die gewünschte Fläche abzubilden. Das könnte in Spezialanwendungen auch so weit gehen, daß eine besonders komplizierte und nicht durch wenige Parameter beschreibbare Oberfläche rein programmtechnisch umgesetzt wird.

Schnittkanten mit anderen Flächen können als Spezialfall betrachtet oder allgemein nach der in 4.2.5 beschriebenen iterativen Methode behandelt werden.

Bei den hier betrachteten Beispielen handelt es sich um solche, deren Startnetz jeweils gleichmäßig verfeinert wurde. Die Behandlung von Punkten auf gekrümmten Flächen ist jedoch nicht daran gebunden. Sie kann bei jeder anderen Verfeinerungsstrategie ebenso durchgeführt werden, da immer nur der aktuelle Näherungspunkt auf eine gegebene Fläche oder Schnittkurve abzubilden ist.

Literatur

[1] T. Apel. SPC-PM Po 3D — User's manual. Preprint SPC 95_33, TU Chemnitz-Zwickau, December 1995.

[2] T. Apel, G. Haase, A. Meyer, and M. Pester. Parallel solution of finite element equation systems: efficient inter-processor communication. Preprint SPC 95_5, TU Chemnitz-Zwickau, Februar 1995.

[3] T. Apel, F. Milde, and M. Theß. SPC-PM Po 3D — Programmer's manual. Preprint SPC 95_34, TU Chemnitz-Zwickau, December 1995.

[4] J. Bey. Der BPX-Vorkonditionierer in 3 Dimensionen: Gitter-Verfeinerung, Parallelisierung und Simulation. Preprint 92 - 03, IWR Universität Heidelberg, 1992.

[5] P. Ciarlet. *The finite element method for elliptic problems.* North–Holland, Amsterdam, 1978.

[6] G. Globisch. PARMESH - a parallel mesh generator. Preprint SPC 93_3, TU Chemnitz-Zwickau, June 1993. Parallel Computing 21, No. 3, March 1995, pp. 509-524.

[7] G. Globisch. On an automatically parallel generation technique for tetrahedral meshes. Preprint SPC 94_6, TU Chemnitz-Zwickau, April 1994.

[8] T. Grund and K. Pietsch. Gitterverfeinerung krummlinig begrenzter Körper. Programmierpraktikum, TU Chemnitz-Zwickau, 1997.

[9] F. Kickinger. Automatic mesh generation for 3D objects. Technical Report No 96-1, Johannes Kepler Universität Linz, Institut f r Mathematik, February 1996.

[10] A. Meyer. A parallel preconditioned conjugate gradient method using domain decomposition and inexact solvers on each subdomain. *Computing*, 45 : 217–234, 1990.

[11] M. Pester. Grafik-Ausgabe vom Parallelrechner für 2D-Gebiete. Preprint SPC 94_24, TU Chemnitz-Zwickau, November 1994.

[12] M. Pester. On-line visualization in parallel computations. In W. Borchers, G. Domick, D. Kröner, R. Rautmann, and D. Saupe, editors, *Visualization Methods in High Performance Computing and Flow Simulation*, pages 91–98, Utrecht, The Netherlands, 1996. VSP. Preprint: SPC 94_23 (Nov. 1994).

[13] U. Reichel. Partitionierung von Finite-Elemente-Netzen. Preprint SFB393/96-18, TU Chemnitz-Zwickau, November 1996.

[14] A. Wierse and M. Rumpf. GRAPE Eine objektorientierte Visualisierungs- und Numerikplattform. *Informatik Forsch. Entw.*, 7:145–151, 1992.

[15] O. C. Zienkiewicz. *Die Methode der finiten Elemente.* Fachbuchverlag, Leipzig, 1975.

Other titles in the SFB393 series:

96-01 V. Mehrmann, H. Xu. Chosing poles so that the single-input pole placement problem is well-conditioned. Januar 1996.

96-02 T. Penzl. Numerical solution of generalized Lyapunov equations. January 1996.

96-03 M. Scherzer, A. Meyer. Zur Berechnung von Spannungs- und Deformationsfeldern an Interface-Ecken im nichtlinearen Deformationsbereich auf Parallelrechnern. March 1996.

96-04 Th. Frank, E. Wassen. Parallel solution algorithms for Lagrangian simulation of disperse multiphase flows. Proc. of 2nd Int. Symposium on Numerical Methods for Multiphase Flows, ASME Fluids Engineering Division Summer Meeting, July 7-11, 1996, San Diego, CA, USA. June 1996.

96-05 P. Benner, V. Mehrmann, H. Xu. A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils. April 1996.

96-06 P. Benner, R. Byers, E. Barth. HAMEV and SQRED: Fortran 77 Subroutines for Computing the Eigenvalues of Hamiltonian Matrices Using Van Loans's Square Reduced Method. May 1996.

96-07 W. Rehm (Ed.). Portierbare numerische Simulation auf parallelen Architekturen. April 1996.

96-08 J. Weickert. Navier-Stokes equations as a differential-algebraic system. August 1996.

96-09 R. Byers, C. He, V. Mehrmann. Where is the nearest non-regular pencil? August 1996.

96-10 Th. Apel. A note on anisotropic interpolation error estimates for isoparametric quadrilateral finite elements. November 1996.

96-11 Th. Apel, G. Lube. Anisotropic mesh refinement for singularly perturbed reaction diffusion problems. November 1996.

96-12 B. Heise, M. Jung. Scalability, efficiency, and robustness of parallel multilevel solvers for nonlinear equations. September 1996.

96-13 F. Milde, R. A. Römer, M. Schreiber. Multifractal analysis of the metal-insulator transition in anisotropic systems. October 1996.

96-14 R. Schneider, P. L. Levin, M. Spasojević. Multiscale compression of BEM equations for electrostatic systems. October 1996.

96-15 M. Spasojević, R. Schneider, P. L. Levin. On the creation of sparse Boundary Element matrices for two dimensional electrostatics problems using the orthogonal Haar wavelet. October 1996.

96-16 S. Dahlke, W. Dahmen, R. Hochmuth, R. Schneider. Stable multiscale bases and local error estimation for elliptic problems. October 1996.

96-17 B. H. Kleemann, A. Rathsfeld, R. Schneider. Multiscale methods for Boundary Integral Equations and their application to boundary value problems in scattering theory and geodesy. October 1996.

96-18 U. Reichel. Partitionierung von Finite-Elemente-Netzen. November 1996.

96-19 W. Dahmen, R. Schneider. Composite wavelet bases for operator equations. November 1996.

96-20 R. A. Römer, M. Schreiber. No enhancement of the localization length for two interacting particles in a random potential. December 1996. to appear in: Phys. Rev. Lett., March 1997

96-21 G. Windisch. Two-point boundary value problems with piecewise constant coefficients: weak solution and exact discretization. December 1996.

96-22 M. Jung, S. V. Nepomnyaschikh. Variable preconditioning procedures for elliptic problems. December 1996.

97-01 P. Benner, V. Mehrmann, H. Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix or Breaking Van Loan's curse? January 1997.

97-02 B. Benhammouda. Rank-revealing 'top-down' ULV factorizations. January 1997.

97-03 U. Schrader. Convergence of Asynchronous Jacobi-Newton-Iterations. January 1997.

97-04 U.-J. Görke, R. Kreißig. Einflußfaktoren bei der Identifikation von Materialparametern elastisch-plastischer Deformationsgesetze aus inhomogenen Verschiebungsfeldern. March 1997.

97-05 U. Groh. FEM auf irregulären hierarchischen Dreiecksnetzen. March 1997.

97-06 Th. Apel. Interpolation of non-smooth functions on anisotropic finite element meshes. March 1997

97-07 Th. Apel, S. Nicaise. The finite element method with anisotropic mesh grading for elliptic problems in domains with corners and edges.

97-08 L. Grabowsky, Th. Ermer, J. Werner. Nutzung von MPI für parallele FEM-Systeme. March 1997.

97-09 T. Wappler, Th. Vojta, M. Schreiber. Monte-Carlo simulations of the dynamical behavior of the Coulomb glass. March 1997.

The complete list of current and former preprints is available via
`http://www.tu-chemnitz.de/sfb393/sfb97pr.html`.