Technische Universität Chemnitz

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Dag Lohse

Ein Standard-File für 3D-Gebietsbeschreibungen - Definition des Fileformats V 2.1 -

Preprint SFB393/98-11

Preprint-Reihe des Chemnitzer SFB 393

SFB393/98-11 **April 1998**

Author's addresses:

Dag Lohse
TU Chemnitz-Zwickau
Fakultät für Mathematik
D-09107 Chemnitz

mail: dlohse@mathematik.tu-chemnitz.de
http://www.tu-chemnitz.de/sfb393/

Inhaltsverzeichnis

1 Motivation

Die Standard–File–Struktur wurde entwickelt, um eine einheitliche Datenschnittstelle zwischen verschiedenen Programmen der Forschergruppe bereitzustellen. Ausgehend vom Aufbau des 2D-Standard–Files wurde eine Datenstruktur erarbeitet, die es ermöglicht, flexibel und plattformunabhängig die Topologie und Geometrie von geometrischen Körpern in Randrepräsentation[1] abzubilden und weiterhin Materialdaten und Randbedingungen für die Lösung von Differentialgleichungen auf diesen 3D–Gebieten aufzunehmen.

Die Standard–File–Struktur ist so konzipiert, daß sie jederzeit änder– und erweiterbar ist, so daß sie wechselnden Anforderungen gerecht werden kann. Um die dabei entstehenden Versionen unterscheiden zu können, wird vom Standard–File ein Versionskennung mitgeführt, an dem die Einleseroutinen erkennen können, wie die Datenzeilen zu interpretieren sind. Alle vorherigen Versionen behalten also Gültigkeit.

Neben der Definition eines Fileformates, die den Aufbau und die Eigenschaften eines Standard–Files beschreibt, gehören zum Umfang der Entwicklung einer einheitlichen Beschreibung der Eingabedaten auch eine Datenbasis, die in der Lage ist, den Inhalt eines Standard–Files im Speicher eines Rechners darzustellen sowie die zur Erzeugung und Manipulation dieser Datenbasis notwendigen Funktionen.

Durch den Einsatz unterschiedlicher Rechnerarchitekturen und mehrerer Programmiersprachen muß diese Datenbasis mit möglichst einfachen Mitteln auskommen. Einfache Mittel heißt in diesem Fall, daß weder strukturierte Datentypen, wie sie in C üblich sind, noch Zeiger, d.h. Speicheradressen, Verwendung finden durften. Da die F77-Programme, für deren Datenbereitstellung das Standard–File entwurfen wurde, grundsätzlich mit einem linearen Feld arbeiten, das den maximal verfügbaren freien Speicherplatz der Maschinen überdeckt, mußte bei dem Einsatz in einer F77–Umgebung auf eine dynamische Speicherverwaltung und auf den Einsatz von Funktionen verzichtet werden, die intern dynamischen Speicher anfordern, das umfaßt z.B. die gepufferten IO-Funktionen von C.

Da die Anwenderprogramme oft nur Teile der Datenbasis benötigen und an die Reihenfolge und den Aufbau der einzelnen Datenfelder oft spezielle Anforderungen stellen, wurde eine Schnittstelle geschaffen, die mit Hilfe einer Anforderungssprache flexibel die benötigten Informationen in dem gewünschten Format aus einem Standard–File extrahiert.

Zum Bearbeiten der Eigenschaften von Gebieten und der Visualisierung von Standard–Files stehen selbständige Programme zur Verfügung. Alle Programme und Funktionen zur Bearbeitung von Standard–Files benutzen dabei zunächst die Datenbasis und die darauf operierenden Funktionen. So sind Änderungen der Definition des Standard–Files schneller durchführbar.

[1] Randrepräsentation: Der Körper wird durch eine Hierarchie seiner begrenzenden Flächen, Kanten und Eckpunkte dargestellt. Diese Darstellung der Topologie des Objekts ist relativ unabhängig von dessen tatsächlichen geometrischen Dimensionen.

Dieses Dokument beschreibt die Definition des Fileformates, d.h. den grundlegenden Aufbau, die Eigenschaften und eine Klassifizierung der möglichen Fehlerzustände.

Die aktuelle Version ist die Version 2.1, die etwa der Version 2.0 entspricht, jedoch einige Veränderungen der optionalen Parameter enthält.

2 Grundaufbau und Eigenschaften

2.1 Ein einführendes Beispiel

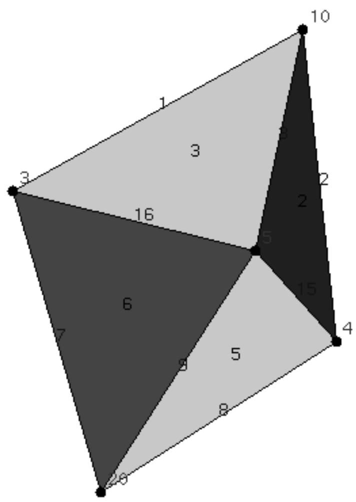

Abbildung 1: Pyramide aus zwei Tetraedern

Das Listing in Abbildung 2 ist eine Repräsentation der Pyramide aus Abbildung 1 in einem Standard–File.

Die beiden Tetraeder (`1` – Zeile *27* und `2` – Zeile *52*) bestehen aus unterschiedlichen Materialien, deren Parameter in den Zeilen *54* und *55* angegeben sind. Sie besitzen eine gemeinsame Fläche, die Fläche mit dem **Namen** `17`, deren Topologie (und Lage) in den Zeilen *15* bis *24* beschrieben wird.

An zwei Flächen (`1` und `2`) befinden sich Neumann–Randbedingungen (beschrieben in den Zeilen *60* bis *63*); an einer Fläche (`6`) sind Dirichlet–Bedingungen vorgegeben (Zeile *57* und Zeile *58*). Die Anzahl der Freiheitsgrade beträgt eins (Zeile *10*: `#DEG_OF_FREE: 1`).

Die Zeile *1* besagt, daß das File in der aktuellen Version (`VERSION: 2.1`) vorliegt. Die Zeilen *2* bis *9* enthalten zusätzliche Informationen, die beim Einlesen dem Nutzer angezeigt werden. Dabei bilden jedoch die Zeilen *3* und *5* eine Ausnahme, da diese nur Kommentarzeilen (`##`) sind. Die Zeile *6* bildet zusammen mit der Zeile *7* **eine logische** Zeile.

```
1  #VERSION: 2.1
2  #DESCRIPTION: Doppelpyramide
3  ##DATE: Mo., 19. Jan. 1998, 08:32:56
4  #DATE: Mon Jan 19 10:23:46 1998
5  ##USER: Dag Lohse
6  #USER: dlohse@galois (HP-UX) (Dag \
7  Lohse, , , )
8  #EQN_TYPE: unknown
9  #PROGRAM: unknown (using bd2) 0.0 pl0
10 #DEG_OF_FREE:      1
11 #HEADER:       8
12   5   9   7   2   0   1   2   2
13 ##V   E   F   S   R   D   N   M   FG
14 ## Die Interface-Flaeche
15 #VERTEX:       3
16   3  -0.500000 -0.333333  0.00000
17   4   0.500000 -0.333333  0.00000
18   5   0.00000   0.666667  0.00000
19 #EDGE:         3
20  14    1       4     3
21  15    1       5     4
22  16    1       3     5
23 #FACE:         1
24  17    1    3     14    15    16
25 ## Koerper 1
26 #SOLID:        1
27   1    1    4      1     2     3    17
28 #FACE:         3
29   1    1    3      1     2    14
30   2    1    3      2     3    15
31   3    1    3      3     1    16
32 #EDGE:         3
33   1    1       10     3
34   2    1       10     4
35   3    1       10     5
36 #VERTEX:       1
37  10   0.00000   0.00000   0.700000
38 ##
39 ## Koerper 2
40 ##
41 #VERTEX:       1
42  20   0.00000   0.00000  -0.700000
43 #EDGE:       3
44   7    1       20     3
45   8    1       20     4
46   9    1       20     5
47 #FACE:       3
48   4    1    3      7     8    14
49   5    1    3      8     9    15
50   6    1    3      9     7    16
51 #SOLID:      1
52   2   42    4      4     5     6    17
53 #MATERIAL:      2
54   1    2       1.00000    1.20000
55  42    1       2.00000
56 #DIRICHLET:      1
57   6
58   1      2.10000
59 #NEUMANN:      2
60   1
61   1      1.10000
62   2
63   1      1.10000
64 #END_OF_DATA:
```

Abbildung 2: Einführendes Beispiel (`doppel.std`)

Die Zeile *11* (`#HEADER: <count>`) trennt den Informationsteil vom Datenteil des Standard–Files und leitet den `#HEADER:`–Block ein, der die maximale Gesamtanzahl von Daten in den einzelnen Blöcken des entsprechenden Datenbereiches angibt. Die `5` an erster Stelle sagt beispielsweise aus, daß maximal fünf Eckpunkte in diesem File beschrieben werden.

Die Zeilen *26 – 37* bzw. die Zeilen *41 – 52* beschreiben die Geometrie und Topologie der beiden Teilkörper. Die Datenblöcke können sich innerhalb des Datenteils wiederholen, ihre Reihenfolge ist ohne Bedeutung. Das Standard–File endet mit der Zeile `#END_OF_DATA:` (*64*).

2.2 Eigenschaften

Um seine Aufgabe als programmübergreifendes Speicherformat erfüllen zu können, muß das Standard–File folgende Eigenschaften haben:

- (für Menschen) lesbar
- verständliche Struktur

- leichte Auffindbarkeit von Datenfehlern
- modifizierbar
- erweiterbar
- systemübergreifend portierbar
- von der Programmiersprache unabhängig

Da das Standard–File nicht für die Speicherung von Feinnetzen eingesetzt werden soll, treten Speicherbedarf und Einlesegeschwindigkeit hinter die oben genannten Eigenschaften zurück.

2.3 Grundaufbau

Um die im Abschnitt 2.2 genannten Eigenschaften zu erfüllen, ist das Standard–File ein zeilenorientiertes (7-bit) ASCII–File, das aus einer Abfolge von Datenblöcken besteht, die durch selbsterklärende Schlüsselworte eingeleitet werden.

Zeilenorientiert ist dabei in dem Sinne zu verstehen, daß jeder Eintrag genau eine **logische** Zeile umfaßt, die jedoch aus mehreren **physischen** Zeile bestehen kann. Zwei physische Zeilen werden durch das betriebssystemübliche Zeilenendezeichen getrennt. Werden mehrere physische Zeichen zu einer logischen Zeile zusammengefaßt, so wird dieses Zeilenendezeichen durch ein `\` (backslash) maskiert. Die Zeilen *6* und *7* in der Abbildung 2 bilden eine solche logische Zeile.

Im Standard–File werden grundsätzlich drei Arten von Zeilen unterschieden, wobei führende Leerzeichen und Tabulatoren ignoriert werden:

- Datenzeilen
- Kommentarzeile (beginnen mit `##`)
- Schlüsselwortzeilen (beginnen mit `#`)

Zwischen diesen Zeilen können beliebig viele Leerzeilen (nur aus Leerzeichen, Tabulatoren und Zeilenendezeichen bestehend) vorkommen. Datenzeilen beginnen mit einer Ziffer, einem Dezimalpunkt oder einem Vorzeichen. Datenzeilen gleicher Struktur werden zu Datenblöcken zusammengefaßt.

Das Standard–File selbst hat folgenden Aufbau:

- Versionskennung: `#VERSION: <version_key>`
- Informationsteil
- Headerblock: `#HEADER: <count>`
- Datenteil
- Fileendemarkierung: `#END_OF_DATA:`

Der Informationsteil enthält Daten, die den Inhalt des Standard–Files und Besonderheiten der repräsentierten Objekte beschreiben. Innerhalb des Informationsteiles können nur Kommentarzeilen und Schlüsselwortzeilen vorkommen. Bei den Schlüsselwortzeilen des Informationsteiles werden wiederum zwei Arten unterschieden:

- Informationszeilen
- Parameterzeilen

Die Informationszeilen enthalten Informationen, die dem Nutzer helfen, das Objekt zu identifizieren. Für den Datenteil sind die Informationszeilen bedeutungslos.
Die Parameterzeilen enthalten Informationen über die zu erwartende Datenmenge und die Struktur der Datenblöcke im Datenteil. Mit Hilfe dieser Parameter ist das Einleseprogramm in der Lage, den Speicherbedarf für die repräsentierten Objekte besser zu schätzen.

Der Datenteil besteht aus Datenblöcken. Ein Datenblock setzt sich aus einer Schlüsselwortzeile (Block-Kopfzeile) und mehreren Datensätzen zusammen.
Ein Datensatz ist entweder eine einzelne Datenzeile oder ein Unterblock, der seinerseits aus einer beschreibenden Kopfzeile und einer oder mehrerer Datenzeilen besteht. Informationszeilen und Parameterzeilen können im Datenteil nicht vorkommen.

Die Beziehungen zwischen den Datenblöcken wird über Namen hergestellt, die aus dem Bereich der positiven ganzen Zahlen frei gewählt werden können und durch eine `INTEGER`-Zahl darstellbar sein müssen. Eine Erweiterung des Namensraumes auf Zeichenketten ist zwar möglich, jedoch zur Zeit nicht vorgesehen.

Die durch das mehrfache Auftreten von Zählern erzeugte Redundanz dient einerseits der Fehlererkennung, wird jedoch andererseits von Umgebungen benötigt, die keine dynamische Speicherverwaltung zulassen.

2.4 Übersicht der Schlüsselwortzeilen

Trennende Schlüsselwortzeilen

`#VERSION:`	Versionskennung Beginn des Informationsteils
`#HEADER:`	Maximale Größe der Datenblöcke trennt Informationsteil und Datenteil
`#END_OF_DATA:`	Ende des Standard–Files

Infomationszeilen

`#DESCRIPTION:`	Kurzbeschreibung der Objekte
`#DATE:`	Erstellungsdatum des Standard–Files
`#USER:`	Ersteller des Standard–Files
`#EQN_TYPE:`	Zugrundeliegende Differentialgleichung
`#DIMENSION:`	geometrische Dimension der Objekte
`#PROGRAM:`	Kennzeichen des Erstellungsprogramms

Parameterzeilen

`#DEG_OF_FREE:`	Anzahl der Freiheitsgrade
`#AVG_DOF_DATA:`	Durchschnittliche Datenanzahl pro Freiheitsgrad
`#AVG_EDGE_DATA:`	Durchschnittliche Datenanzahl pro Kante (nur wegen Vollständigkeit)
`#AVG_FACE_DATA:`	Durchschnittliche Datenanzahl pro Fläche
`#AVG_SOLID_DATA:`	Durchschnittliche Datenanzahl pro Körper
`#AVG_REGION_DATA:`	Durchschnittliche Datenanzahl pro Gebiet
`#AVG_FACE_GEO_DATA:`	Durchschnittliche Datenanzahl pro Flächengeometriebeschreibung
`#AVG_MATERIAL_DATA:`	Durchschnittliche Datenanzahl pro Materialbeschreibung

Block-Kopfzeilen

`#VERTEX:`	Datenblock der Eckpunktdaten
`#EDGE:`	Datenblock der Kantendaten
`#FACE:`	Datenblock der Flächendaten
`#SOLID:`	Datenblock der Körperdaten
`#REGION:`	Datenblock der Gebietsdaten
`#MATERIAL:`	Datenblock der Materialdaten
`#FACE_GEO:`	Datenblock der Flächengeometrie
`#NEUMANN:`	Datenblock der Neumann–Randbedingungen
`#DIRICHLET:`	Datenblock der Dirichlet–Randbedingungen

3 Informationszeilen und Versionskennung

3.1 #VERSION: <versionskennung>

Die `#VERSION:`–Zeile leitet **jedes** Standard–File ein. Durch sie wird festgelegt, welche Schlüsselwortzeilen im File auftreten dürfen und wie die Datenzeilen zu interpretieren sind.

`<versionskennung>` ist eine Zeichenkette, die als Versionskennung eindeutig festlegt, welche Version des Standard–Files vorliegt. Zur Zeit sind folgende Versionskennungen erlaubt:

- `1.0`
- `2.0`
- `2.1`

Die Unterschiede der einzelnen Versionen werden im Abschnitt 6 näher erläutert.

3.2 #DESCRIPTION: <beschreibung>

Die Zeichenkette `<beschreibung>` dient der Aufnahme einer **kurzen** aber informativen Beschreibung der Objekte, die durch das Standard–File repräsentiert werden. Sie sollte keine Angaben enthalten, die in anderen Informationszeilen abgelegt sind (Nutzername oder Programm).

3.3 #DATE: <datum>

Die Zeichenkette `<datum>` ist für die Aufnahme des Erstellungsdatums vorgesehen. Da keinerlei Datumsformat festgelegt ist, sollten Katalogisierungstools, die auf diese Information zurückgreifen, in der Lage sein, `<datum>` flexibel auszuwerten.

3.4 #EQN_TYPE: <gleichunstyp>

Die Zeichenkette `<gleichunstyp>` dient zur Beschreibung der Differentialgleichung, für die die Randbedingungen und Materialdaten des Files vorgesehen sind. Diese Zeile wird zur Zeit noch nicht automatisch ausgewertet, sie dient lediglich der Information des Nutzers. Der Inhalt von `<gleichunstyp>` ist noch nicht standardisiert.

3.5 #DIMENSION: <dimension>

Die Zeichenkette `<dimension>` wird zur Zeit nicht ausgewertet, sie ist dazu vorgesehen, die Speicherung von 2D-Objekten in Standard–Files zu ermöglichen.

3.6 #PROGRAM: <pgm-name>

Die Zeichenkette `<pgm-name>` sollte neben dem Namen des Programmes, mit dem das Standard–File erzeugt wurde, auch Informationen über die Programmversion und das Betriebssystem enthalten, um die Fehlersuche zu erleichtern.

4 Parameterzeilen und #HEADER–Block

4.1 #DEG_OF_FREE: <dof>

`<dof>`	**Wertebereich:** `0 -- MAXINT`	**Default:** `3`

`<dof>` ist ein INTEGER–Wert, der angibt, wieviele Freiheitsgrade für die Randbedingungen einer Fläche vorgesehen sind. Mit diesem Wert wird gleichzeitig festgelegt, wieviele Datensätze in einem Unterblock der Randbedingungen vorhanden sind. Ein solcher Unterblock **muß** genau `<dof>` Datenzeilen besitzen. Werden nur einzelne Freiheitsgrade an dieser Fläche benötigt, so sind die restlichen Freiheitsgrade trotzdem als Datenzeile anzugeben. Diese sind dann als ungenutzt zu markieren. Siehe dazu auch Abschnitt 5.7.

4.2 #AVG_DOF_DATA: <v-count>

`<count>`	**Wertebereich:** `0 -- MAXINT`	**Default:** `5`

Der INTEGER–Wert `<v-count>` gibt die durchschnittliche Anzahl der REAL–Einträge einer Datenzeile eines Datenblockes der Randbedingungen an. Siehe dazu auch Abschnitt 5.7.

4.3 #AVG_EDGE_DATA: <v-count>

`<v-count>`	**Wertebereich:** `2 -- MAXINT`	**Default:** `2`

`#AVG_EDGE_DATA:` wird zur Zeit nicht ausgewertet. In späteren Versionen des Standard–Files, die neben den beiden Begrenzungspunkten einer Kante noch die Angabe weiterer Punkte erlauben (Mittelpunkte, Unterteilungspunkte etc.) nimmt `<v-count>` (INTEGER) die durchschnittliche Anzahl der Punktnamen einer Kante auf.

4.4 #AVG_FACE_DATA: <e-count>

`<e-count>`	**Wertebereich:** `3 -- MAXINT`	**Default:** `5`

Der INTEGER–Wert `<e-count>` gibt an, wieviele Kanten durchschnittlich zu einer Fläche gehören. Werden komplexe Geometrien in einem Standard–File gespeichert, so muß dieser Wert erhöht werden, werden jedoch viele einfache Flächen gespeichert, wie sie z.B. bei Netzen auftreten, so kann der Wert `<e-count>` entsprechend verringert werden, um den Speicherbedarf der Einleseroutinen zu verringern.

4.5 #AVG_SOLID_DATA: <f-count>

`<f-count>`	**Wertebereich:** `4 -- MAXINT`	**Default:** `berechnet`

`<f-count>`, ein INTEGER–WERT, gibt an, wieviele Flächen durchschnittlich zur Definition eines Körpers benutzt werden. Mit einer günstigen (realistischen) Wahl dieses Parameters läßt sich der Speicherbedarf der Einleseroutinen ebenso verringern wie in Abschnitt 4.4 beschrieben. Bei Quadernetzen ist dieser Wert beispielsweise auf 6 zu setzten.

4.6 #AVG_REGION_DATA: <s-count>

`<s-count>`	**Wertebereich:** `0 -- MAXINT`	**Default:** `berechnet`

Der INTEGER–Wert `<s-count>` gibt an, wieviele Körper durchschnittlich zu einem Gebiet zusammengefaßt werden. Zur Wahl dieses Parameters gelten die Ausführungen aus Abschnitt 4.4 analog.

4.7 #AVG_FACE_GEO_DATA: <count>

<count>	**Wertebereich:** 0 -- MAXINT	**Default:** 8

<count>, ein INTEGER–Wert, gibt an, wieviele REAL–Werte durchschnittlich benötigt werden, um eine Flächengeometrie zu beschreiben. Ein vom Defaultwert abweichender Wert ist nur sinnvoll, wenn neue Typen der Flächengeometrie eingesetzt werden.

4.8 #AVG_MATERIAL_DATA: <count>

<count>	**Wertebereich:** 0 -- MAXINT	**Default:** 10

Der INTEGER–Wert <count> gibt an, wieviele REAL–Daten durchschnittlich für die Beschreibung eines Material–Datensatzes benötigt werden.

4.9 #HEADER: <count>

<count>	**Wertebereich:** 4 -- 9	**Default:** -

Die #HEADER:–Zeile trennt den Informationsteil des Standard–Files von dessen Datenteil. Nach der #HEADER:–Zeile darf keine Informations– oder Parameterzeile auftreten.

Die #HEADER:–Zeile ist die Block-Kopfzeile des HEADER–Blocks, der die Information über die Größe der Datenblöcke enthält. Der HEADER–Block besteht nur aus einer Zeile, die folgenden Aufbau hat:

<v-c>	<e-c>	<f-c>	<s-c>	<r-c>	<d-c>	<n-c>	<m-c>	<g-c>

Die INTEGER–Zahl <count> in der Block-Kopfzeile gibt an, wieviele dieser Werte gültig sind. Die INTEGER–Werte des #HEADER:–Blocks haben folgende Bedeutung:

<v-c> Maximale Anzahl von Eckpunkten
<e-c> Maximale Anzahl von Kanten
<f-c> Maximale Anzahl von Flächen
<s-c> Maximale Anzahl von Körpern
<r-c> Maximale Anzahl von Gebieten
<d-c> Maximale Anzahl von Dirichlet–Randbedingungen
<n-c> Maximale Anzahl von Neumann–Randbedingungen
<m-c> Maximale Anzahl von Materialdaten
<g-c> Maximale Anzahl von Flächengeometrie–Beschreibungen

Da zur Beschreibung eines Objektes grundsätzlich Eckpunkte, Kanten, Flächen und Körper benötigt werden, sind die ersten vier Einträge im **#HEADER:**–Block notwendig, die anderen Werte müssen nur angegeben werden, wenn entsprechende Datenblöcke vorhanden sind.

Die Werte des **#HEADER:**–Blocks geben nur an, wieviele Datensätze in den Datenblöcken **maximal** vorkommen dürfen, die exakte Anzahl kann daraus nicht abgeleitet werden.

5 Datenblöcke

5.1 #VERTEX: <count>

Der **#VERTEX:**–Block beschreibt die geometrische Lage der (Eck–)Punkte der Objekte. Die INTEGER–Zahl <count> gibt an, wieviele der folgenden Datenzeilen gültig sind. Eine Datenzeile hat folgende Gestalt:

<v-name>	<x-koord>	<y-koord>	<z-koord>

Die Werte haben folgende Bedeutung:

<v-name>	INTEGER	Name des Punktes
<x-koord> <y-koord> <z-koord>	REAL	Koordinaten des Punktes

Obwohl das Standard–File problemlos mehrere Punkte mit gleichen Koordinaten speichern kann, sollte man beachten, daß zur Zeit nur mit der Genauigkeit REAL*4 gearbeitet wird.

5.2 #EDGE: <count>

Der **#EDGE:**–Block beschreibt die **Topologie** der Kanten, indem jede Kante durch Namen ihrer Eckpunkte definiert wird. Die tatsächliche geometrische Gestalt der Kante ergibt sich implizit aus den Geometrie–Daten der Flächen, die diese Kante bilden. Gehört eine Kante zu mehr als zwei Flächen, so müssen die Geometrie–Daten der beteiligten Flächen entweder die Bildung **einer** Kante zwischen den beiden Punkten zulassen, oder es muß eine zweite Kante, die die gleichen Endpunkte besitzt, definiert werden. Die INTEGER–Zahl <count> gibt an, wieviele der folgenden Datenzeilen im **#EDGE:**–Block gültig sind.

Eine Datenzeile hat folgende Gestalt:

<e-name>	<e-typ>	<s-punkt>	<e-punkt>

Die Werte haben folgende Bedeutung:

<e-name>	INTEGER	Name der Kante
<e-typ>	INTEGER	reserviert (1)
<s-punkt>	INTEGER	Name des Startpunktes
<e-punkt>	INTEGER	Name des Endpunktes

Der Wert von <e-typ> ist für die aktuelle Version des Standard–Files ohne Bedeutung, sollte jedoch stets mit 1 angegeben werden. Er kann später dazu dienen, Kanten zu markieren, die spezielle Eigenschaften haben. Eine solche Eigenschaft könnte beispielsweise das Mitführen einer Punktliste zur Unterteilung der Kante sein. Die Namen dieser Punkte könnten dann nach den Endpunkten der Kante angegeben werden. In der aktuellen Version des Standard–Files werden jedoch solche Punkte ignoriert, falsche Werte bei <e-typ> lösen eine Fehlermeldung aus.

Es ist vorgesehen, daß der #EDGE:–Block durch den Inhalt der #AVG_EDGE_DATA:–Zeile beeinflußt wird.

5.3 #FACE: <count>

Der #FACE:–Block beschreibt die Topologie einer Fläche und enthält Verweise auf die geometrischen Eigenschaften dieser Flächen. Eine Fläche wird durch die Angabe der Namen aller Kanten definiert, die zu ihr gehören. Die Reihenfolge der Kanten ist dabei ohne Bedeutung, entartete (selbstüberlappende) Flächen sind im Standard–File unzulässig. Der INTEGER–Wert <count> gibt an, wieviele gültige Datensätze im #FACE:–Block enthalten sind.

Eine Datenzeile hat folgende Gestalt:

<f-name>	<fg-ptr>	<e-count>	<e-1>	...	<e-k>

Die Werte haben folgende Bedeutung:

<f-name>	INTEGER	Name der Fläche
<fg-ptr>	INTEGER	Name der Flächen-Geometrie
<e-count>	INTEGER	Anzahl der definierenden Kanten
<e-1>	INTEGER	Name der ersten Kante
<e-k>	INTEGER	Name der letzten (k = <e-count>) Kante

Der Wert <fg-ptr> ist ein Name einer Flächengeometrie–Beschreibung, die im #FACE_GEO–Block abgelegt ist. Dabei hat der Wert 1 eine Sonderbedeutung, da dieser das Kennzeichen dafür ist, daß die betreffende Fläche **keine** besonderen geometrischen Eigenschaften besitzt, d.h. voll in einer Ebene liegt. Ist im Standard–File kein #FACE_GEO–Block vorgesehen (d.h. auch im #HEADER:–Block steht eine 0), so müssen die Werte <fg-ptr> vom Anwenderprogramm interpretiert werden.

<e-count> gibt an, wieviele Kantennamen zur Definition der Fläche folgen. Die Kanten <e-1> ... <e-k> müssen ein geschlossenes, zusammenhängendes, doppelpunkfreies Polygon bilden. Dabei ist zu beachten, daß Punkte, die zwar gleiche Koordinaten besitzen, jedoch unterschiedliche Namen tragen, als verschieden angesehen werden.

Der #FACE:–Block wird einerseits von der #AVG_FACE_DATA:–Zeile und andererseits vom Vorhandensein eines #FACE_GEO:–Blockes beeinflußt.

5.4 #SOLID: <count>

Der #SOLID:–Block beschreibt die Topologie der Körper und enthält Verweise auf die Materialdaten. Ein Körper wird durch die Angabe der Namen aller Flächen definiert, die ihn begrenzen. Die Reihenfolge der Flächen ist dabei ohne Bedeutung, sie müssen lediglich ein (einfach zusammenhängendes) Polyeder bilden. Der INTEGER–Wert <count> gibt an, wieviele gültige Datensätze im #SOLID:–Block enthalten sind.

Eine Datenzeile hat folgende Gestalt:

<s-name>	<mat-ptr>	<f-count>	<f-1>	...	<f-k>

Die Werte haben folgende Bedeutung:

<s-name>	INTEGER	Name des Körpers
<mat-ptr>	INTEGER	Name des Materials
<f-count>	INTEGER	Anzahl der begrenzenden Flächen
<f-1>	INTEGER	Name der ersten Fläche
<f-k>	INTEGER	Name der letzten (k = <f-count>) Fläche

Der Wert <mat-ptr> ist der Name eines Materialdatensatzes, der im #MATERIAL:–Block abgelegt ist. Eine 0 markiert Körper, für die keine Materialdaten gegeben sind. Fehlt der #MATERIAL:–Block im Standard–File vollständig (also auch im HEADER:–Block), so ist der Nutzer selbst für die Interpretation dieses Wertes verantwortlich.

Der Wert <f-count> gibt an, wieviele Flächen den Körper begrenzen, <f-1> bis <f-k> sind die Namen dieser Flächen. Dabei ist zu beachten, daß zwei Kanten nur dann als gleich betrachtet werden, wenn sie den gleichen Namen tragen, die Übereinstimmung der Topologie reicht dazu nicht aus.

Der #SOLID:–Block wird einerseits von der #AVG_SOLID_DATA:–Zeile und andererseits vom Vorhandensein eines #MATERIAL:–Blockes beeinflußt.

5.5 #REGION: <count>

Der `#REGION:`–Block faßt mehrere Körper zu einem Gebiet zusammen. Dieses Gebiet kann dann von Nutzer mit bestimmten Eigenschaften versehen werden.

Der INTEGER–Wert `<count>` gibt an, wieviele gültige Datensätze im `#REGION:`–Block enthalten sind.

Eine Datenzeile hat folgende Gestalt:

`<r-name>`	`<r-type>`	`<s-count>`	`<s-1>`	...	`<s-k>`

Die Werte haben folgende Bedeutung:

`<r-name>`	INTEGER	Name des Gebietes
`<r-type>`	INTEGER	Typ des Gebietes
`<s-count>`	INTEGER	Anzahl der zugehörigen Körper
`<s-1>`	INTEGER	Name des ersten Körpers
`<s-k>`	INTEGER	Name der letzten (`k = <s-count>) Körpers`

Der `#REGION:`–Block hat zur Zeit kein spezielles Einsatzgebiet, es steht jedoch dem Nutzer frei, spezielle Zusammenfassungen von Körpern zu definieren.

Der `#REGION:`–Block wird durch die `#AVG_REGION_DATA:`–Zeile beeinflußt.

5.6 #DIRICHLET: <count>

Der `#DIRICHLET:`–Block beschreibt Dirichlet–Randbedingungen einzelner Flächen. Dabei besitzen alle Flächen, die überhaupt Randbedingungen tragen, die gleiche Anzahl von Freiheitsgraden. Diese Anzahl wird durch die Parameterzeile `#DEG_OF_FREE:` festgelegt, fehlt diese, so wird der Defaultwert 5 benutzt. Der INTEGER–Wert `<count>` gibt die Anzahl von Flächen an, für die gültige Randbedingungs–Datensätze folgen.

Die Datensätze des `#DIRICHLET:`–Blocks sind Unterblöcke, die von einer Kopfzeile eingeleitet werden und jeweils soviele Datenzeilen besitzen, wie Freiheitsgrade vorliegen. Die Kopfzeile enthält nur einen INTEGER–Wert, den Namen der Fläche. Eine Datenzeile des Unterblocks hat folgenden Aufbau:

`<rb-type>`	`<d-1>`	...	`<d-k>`

Die Werte haben folgende Bedeutung:

`<rb-type>`	INTEGER	Gleichungstyp
`<d-1>`	REAL	erster Parameter der Gleichung
`<d-k>`	REAL	letzter Parameter der Gleichung

Jeder Freiheitsgrad kann als eine Gleichung angegeben werden, die durch eine Typangabe in `<rb-type>` verschlüsselt wird. Die Werte von `<rb-type>` verschlüsseln folgende Gleichungen:

`rbtype=0`	Dieser Freiheitsgrad stellt (für diese Fläche) eine offene Randbedingung dar, es werden keine `rbdata` benötigt.
`rbtype=1`	Dieser Freiheitsgrad ist für die gesamte Fläche konstant, der Parameter `rbdata[1]` enthält den Wert dieser Konstanten.
`rbtype=2`	Der Freiheitsgrad wird durch eine lineare Gleichung im *globalen* Koordinatensystem beschrieben. Der Wert $f(x, y, z)$ dieses Freiheitsgrades ergibt sich aus: $f(x, y, z) = \texttt{rbdata[1]} * x + \texttt{rbdata[2]} * y + \texttt{rbdata[3]} * z + \texttt{rbdata[4]}$.

Der `#DIRICHLET:`–Block wird durch die Parameterzeile `#DEG_OF_FREE:` und die Parameterzeile `#AVG_DOF_DATA:` beeinflußt.

5.7 #NEUMANN: <count>

Der `#NEUMANN:`–Block beschreibt die Neumann–Randbedingungen einzelner Flächen. Er hat den gleichen Aufbau wie der `#DIRICHLET:`–Block.

Der `#NEUMANN:`–Block wird durch die Parameterzeile `#DEG_OF_FREE:` und die Parameterzeile `#AVG_DOF_DATA:` beeinflußt.

5.8 #MATERIAL: <count>

Der `#MATERIAL:`–Block beschreibt Materialeigenschaften, auf die im `#SOLID:`–Block bezug genommen wird. `<count>` (INTEGER) gibt an, wieviele Material–Datensätze folgen.

Ein solcher Datensatz hat folgenden Aufbau:

`<m-name>`	`<m-count>`	`<d-1>`	...	`<d-k>`

Die Werte haben folgende Bedeutung:

`<m-name>`	INTEGER	Name des Materials
`<m-count>`	INTEGER	Anzahl der Material–Parameter
`<d-1>`	REAL	erster Parameter des Materials
`<d-k>`	REAL	letzter Parameter des Materials

Der `#MATERIAL:`–Block wird durch die Parameterzeile `#AVG_MATERIAL_DATA:` beeinflußt.

5.9 #FACE_GEO: <count>

Der `#FACE_GEO:`–Block beschreibt die geometrische Gestalt von Flächen. Auf die Daten dieses Blockes wird im `#FACE:`–Block bezug genommen. `<count>` gibt an, wieviele Flächengeometrien in den folgenden Datenzeilen beschrieben werden.

Eine solche Datenzeile hat folgenden Aufbau:

`<fg-name>`	`<fg-typ>`	`<fg-count>`	`<d-1>`	...	`<d-k>`

Die Werte haben folgende Bedeutung:

`<fg-name>`	INTEGER	Name der Geometriebeschreibung
`<fg-typ>`	INTEGER	Code für Gleichungstyp
`<fg-count>`	INTEGER	Anzahl der REAL–Daten
`<d-1>`	REAL	erster Parameter der Geometriebeschreibung
`<d-k>`	REAL	letzter Parameter der Geometriebeschreibung

Dabei ist besonders darauf zu achten, daß der Name 1 nicht als Name für eine Geometriebeschreibung zur Verfügung steht, da ihm eine Sonderbedeutung im `#FACE:`–Block zukommt. Siehe dazu auch Abschnitt 5.3.

Die Codes für den Gleichungstyp müssen folgenden Bereichen entstammen:

1	...	10	Ebene
11	...	20	Zylinderfläche
21	...	30	Kugeloberfläche
31	...	40	Kegelmantel
41	...	50	Rotationsfläche
51	...	60	Torusfläche

Tabelle 1 enthält die bisher definierten Typen.

Der `#FACE_GEO:`–Block wird durch die `#AVG_FACE_GEO_DATA:`–Zeile beeinflußt.

6 Unterschiede zu früheren Versionen

6.1 Parameter

Da beim Übergang von der Version 1.0 auf die Versionen 2.x einige Konzepte verändert wurden, mußten auch entsprechende Änderungen in den Parameterzeilen vorgenommen werden. Bei dieser Gelegenheit wurden wirkungslose Parameter entfernt und versucht konsistente Namen einzuführen.

`<fg-name>`	`<fg-count>`	Beschreibung
1	6	(n_x, n_y, n_z),(x_0, y_0, z_0) Normalenvektor, Punkt auf Ebene
2	6	(x_0, y_0, z_0),(n_x, n_y, n_z) Punkt auf Ebene, Normalenvektor
11	7	(x_0, y_0, z_0),(x_1, y_1, z_1),r zwei Punkte auf Zylinderachse, Radius
21	4	(x_0, y_0, z_0),r Kugelmittelpunkt, Radius
31	7	(x_0, y_0, z_0),(x_1, y_1, z_1),r Kegelspitze, Punkt auf Kegelachse, Radius an dieser Stelle
41	8	(x_0, y_0, z_0),(x_1, y_1, z_1),A,B zwei Punkte auf Rotationsachse, Parameter der Gleichung der rotierenden Kurve $\pm\frac{\xi^2}{a^2} \pm \frac{\mu^2}{b^2} = 1$ mit $A = \pm a^2$, $B = \pm b^2$ Bem.: $B < 0$ einschaliges, $A < 0$ zweischaliges Hyperboloid
51	8	(x_0, y_0, z_0),(n_x, n_y, n_z),R,r Torusmittelpunkt, Normalenvektor der Grundebene, Rotationsradius (Abstand zum Mittelpunkt des rotierenden Kreises), Radius des rotierenden Kreises

Tabelle 1: Vordefinierte Typen von Flächengeometrien

Folgende (bisher wirkungslose) Parameter wurden entfernt:

`#MAX_EDGE_PTR_DATA:`	jetzt `#AVG_EDGE_DATA:`
`#MAX_FACE_PTR_DATA:`	jetzt `#AVG_FACE_DATA:`
`#MAX_SOLID_PTR_DATA:`	jetzt `#AVG_SOLID_DATA:`
`#MAX_REGION_PTR_DATA:`	jetzt `#AVG_REGION_DATA:`
`#MAX_BC_PTR_DATA:`	jetzt `#AVG_DOF_DATA:`

Durch die Änderung der Bedeutung der in der Version 1.0 als Typ–Einträge bezeichneter

Werte verlieren folgende Parameterzeilen ihren Sinn:

#MAX_EDGE_TYPE:	Kantentyp wird ignoriert
#MAX_FACE_TYPE:	Verweis auf #FACE_GEO:–Block
#MAX_SOLID_TYPE:	Verweis auf #MATERIAL:–Block
#MAX_REGION_TYPE:	Gebietstyp wird ignoriert
#MAX_BC_TYPE:	Gleichungstyp legt nicht mehr Speicherbedarf fest

Die Parameterzeile

#MAX_MAT_DATA:	Maximale Anzahl von Daten in Materialbeschreibung

wird zwar noch akzeptiert, sollte aber durch #AVG_MATERIAL_DATA: ersetzt werden.

Folgende Parameterzeilen sind mit der Version 2.0 hinzugekommen:

#AVG_EDGE_DATA:	ersetzt #MAX_EDGE_PTR_DATA: und #MAX_EDGE_TYPE:
#AVG_SOLID_DATA:	ersetzt #MAX_SOLID_PTR_DATA: und #MAX_SOLID_TYPE:
#AVG_REGION_DATA:	ersetzt #MAX_REGION_PTR_DATA: und #MAX_REGION_TYPE:
#AVG_MATERIAL_DATA:	ersetzt #MAX_MAT_DATA:
#AVG_FACE_GEO_DATA:	neu für #FACE_GEO:–Block
#AVG_DOF_DATA:	ersetzt #MAX_BC_PTR_DATA: und #MAX_BC_TYPE:

6.2 Datenblöcke

Beim Übergang von Version 1.0 zu Version 2.0 wurde ein neuer Datenblock (#FACE_GEO:) eingeführt (siehe dazu auch Abschnitt 5.9) und die Bedeutung der in Version 1.0 als Typeinträge bezeichneten Werte grundlegend geändert. Der #HEADER:–Block wurde um einen Eintrag erweitert.

In den Blöcken #EDGE: und #REGION: werden die <typ>–Einträge grundsätzlich ignoriert, sollten jedoch stets den Wert 1 haben.

`#FACE:`

Der zweite Wert einer Datenzeile (`<type>` bzw. `<fg-ptr>`) war in der Version 1 als ein Typanzeiger vorgesehen. Dieses Konzept wurde durch die externe Beschreibung der Flächengeometrie in einem gesonderten Block (`#FACE_GEO:`) abgelöst. Dieser Wert enthält infolgedessen den Namen einer solchen Flächengeometriebeschreibung. Ist jedoch kein `#FACE_GEO:`–Block vorgesehen, d.h. der entsprechende Eintrag im `#HEADER:`–Block ist Null oder fehlt ganz, so werden diese Werte ignoriert, d.h. unverändert dem Nutzer zur Verfügung gestellt. In diesem Falle wird der Nutzer jedoch in einer Warnung darauf hingewiesen, daß keine Flächengeometriebeschreibungen vorhanden sind.

`#SOLID:`

Ähnlich wie im `#FACE:`–Block, war der zweite Wert (`<type>` bzw. `<mat-ptr>`) einer Datenzeile des `#SOLID:`–Blocks in der Version 1 als Typanzeiger. Da das Konzept der Bereitstellung von Materialdaten grundlegend geändert wurde, enthält dieser Wert den Namen einer Materialbeschreibung aus dem `#MATERIAL:`–Block. Sind jedoch keine Materialdaten vorgesehen, d.h. der entsprechende Eintrag im `#HEADER:`–Block ist Null oder fehlt ganz, so werden diese Werte ignoriert, d.h. unverändert dem Nutzer zur Verfügung gestellt. In diesem Falle wird der Nutzer jedoch in einer Warnung darauf hingewiesen, daß keine Materialdaten vorhanden sind, wobei diese Warnung entfällt, wenn alle `<mat-ptr>` den Wert 1 aufweisen.

`#REGION:`

Der `#REGION:`–Block diente in der Version 1 der Zusammenfassung von Körpern, um diesen gemeinsame Materialeigenschaften zuzuweisen. Diese Aufgabe entfällt ab der Version 2.0. Der `#REGION:`–Block kann vom Nutzer jedoch weiter für die Definition beliebiger Gebiete verwendet werden.

`#MATERIAL:`

Ab der Version 2.0 wird ein neues Konzept zur Behandlung von Materialeigenschaften benutzt. In der Version 1.0 wurden Materialien einzelnen Körpern oder, falls vorhanden, einzelnen Gebieten zugeordnet. Ab der Version 2.0 werden die Körper den Materialdaten zugeordnet, d.h. jedes Material trägt einen eindeutigen Namen, auf den sich im `#SOLID:`–Block bezogen werden kann.

Für eine Datenzeile des `#MATERIAL:`–Blocks ergibt sich damit einer Änderung der Bedeutung des ersten Wertes, der in der Version 1.0 den Namen eines Körpers bzw. Gebietes repräsentierte, stellt jetzt den Namen des Materials dar.

`#NEUMANN:` und `#DIRICHLET:`

Die Änderung bei den Datensätzen der Randbedingungen bestehen darin, daß beim Auftreten von unbekannten Gleichungstypen alle nachfolgenden Werte bis zum Ende der logischen Zeile eingelesen werden und nur eine Warnung des Nutzers erfolgt.

6.3 Neuerungen in Version 2.1

Ab Version 2.1 können in allen (Nicht–Kommentar–)Zeilen des Standard–Files an allen Stellen, an denen Leerzeichen vorkommen dürfen, maskierte Zeilenenden vorkommen. Bisher war dies nur für Datenzeilen vorgesehen.

7 Klassen von Datenfehlern

7.1 Kritische Fehler

Kritische Fehler sind Fehler, die ein weiteres Bearbeiten des Standard–Files unmöglich machen, es erfolgt ein sofortiger Abbruch der Einlesefunktion.

Zu den kritischen Fehlern gehören:

- Fehlendes Standard–File,
- Defektes Standard–File,
- Fehlende `#VERSION:`–Zeile,
- Fehlender `#HEADER:`–Block,
- Speicherplatzmangel.

7.2 Schwere Fehler

Schwere Fehler sind Fehler, die zwar prinzipiell das weitere Einlesen des Standard–Files erlauben, jedoch mit hoher Wahrscheinlichkeit zu unbrauchbaren Daten führen. Für diese Fehler wird ein Nutzerdialog durchgeführt, ist dieser abgeschaltet, so werden schwere Fehler wie kritischen Fehler behandelt.

Zu den schweren Fehlern gehören:

- Defekte oder unvollständige Datenzeilen,
- Fehlende Namen in Topologie oder Randbedingungen,
- Mehrfache Namen bei Topologiedaten,
- Namen außerhalb des Namensraumes (1...MAXINT),
- Überschreitung des reservierten Speicherbereiches bei Datenzeilen,
- Fehlerhafter `#HEADER:`–Block.

7.3 Unkritische Fehler

Unkritische Fehler sind Fehler, die zu Folgefehlern führen können, jedoch ein (eingeschränktes) Weiterarbeiten mit den eingelesenen Daten zulassen. Für diese Fehler wird ein Nutzerdialog durchgeführt, ist dieser abgeschaltet, so werden sie wie Warnungen behandelt.

Zu den unkritischen Fehlern gehören:

- Unbekannte Schlüsselwörter,
- Überzählige Datenzeilen,
- Mehrfache Nutzung einer Parameterzeile,
- Nutzung konkurrierender Parameterzeilen,
- Fehlerhafte Parameterzeilen,
- Parameterzeilen im Datenblock,
- Falsche Datenblöcke (im `#HEADER:`–Block nicht oder mit Null angegeben),
- Mehrfache oder fehlende Namen für Material oder Flächengeometrie,

- Mehrfache Namen bei Randbedingungen.
- Unbekannte Versionskennung,
- Überschreitung des reservierten Speicherbereiches bei Informationszeilen.

7.4 Warnungen

Warnungen dienen der Anzeige von Inkonsistenzen des Standard–Files, die nur geringe Auswirkung auf die weitere Bearbeitung des Standard–Files haben, jedoch auch auf Datenfehler hinweisen können.

Warnungen werden hervorgerufen durch:

- Mehrfache Nutzung einer Informationszeile,
- Fehlender `#MATERIAL:`–Block bei Nutzung von Materialnamen (ungleich eins)
- Fehlender `#FACE_GEO:`–Block bei gleichzeitiger Angabe von Flächengeometrie–Namen im `#FACE:`–Block
- Falsche Typen bei Flächengeometriebeschreibungen und Randbedingungen

Index