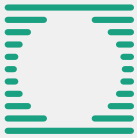
IBJpluris 3.1

Eine Implementierung des Ausbreitungsmodells PLURIS

(Janicke, U., Janicke, L., Atmospheric Environment 35, 2001, 877-890)

Ing.-Büro Janicke, Überlingen

Januar 2019



Ingenieurbüro Janicke
Gesellschaft für Umweltphysik
www.janicke.de

Hermann-Hoch-Weg 1
D-88662 Überlingen
info@janicke.de

Vorbemerkung

Diese Dokumentation bezieht sich auf die Programmversion 3.1.

1 Einführung

Ein Fahnenüberhöhungsmodell beschreibt die Auswirkung der dynamischen Eigenschaften der Abluft (Impuls-, Temperatur- und Feuchtedifferenz zur Umgebungsluft) auf den mittleren Verlauf der Fahnenachse. Nach der Freisetzung wird Umgebungsluft in die Fahne eingemischt, so dass sich die über den Fahnenquerschnitt gemittelten Parameter wie Impuls, Temperatur und Feuchte sowie der Fahnenradius ändern. Dies führt im Vergleich zu einer passiven Freisetzung zu einem anderen Verlauf der Fahnenachse und zu einer Vergrößerung des Fahnenradius.

Das Einmischen wird durch eine sogenannte Einmischfunktion beschrieben, die in gängigen Modellen nur von Differenzgrößen in Bezug auf die Umgebungsluft abhängt. Hiermit wird das Einmischen aufgrund der fahneninduzierten Turbulenz beschrieben, die im Anfangsstadium der Fahne in der Regel deutlich größer ist als die atmosphärische Turbulenz. Wenn die Fahnenparameter sich denen der Umgebungsluft angenähert haben, wird die weitere Aufweitung der Fahne vor allem durch die atmosphärische Turbulenz bestimmt, so wie es in einem Ausbreitungsmodell beschrieben wird.

Es gibt daher einen Übergangsbereich zwischen beiden Modellkonzepten, der dynamischen Beschreibung der Fahne als Störung der atmosphärischen Strömung einerseits und der Modellierung einer passiven Spurenstoffwolke im Rahmen einer Ausbreitungsrechnung andererseits. Die beiden Bereiche können durch das Konzept einer Endüberhöhung miteinander verbunden werden. Hierbei wird ein Kriterium für die Annahme festgelegt, daß die Fahne ihre Eigenständigkeit verloren hat und sich in ihren mechanischen und thermodynamischen Eigenschaften nicht mehr wesentlich von der Umgebungsluft unterscheidet. Bei labiler Schichtung kommt die Überlegung dazu, daß Konvektionszellen (*downdrafts*) die Fahne bis zum Erdboden herunter transportieren können und so einer Überhöhung im Mittel entgegenwirken.

In der Veröffentlichung *A three-dimensional plume rise model for dry and wet plumes* (Janicke, U., Janicke, L., Atmos. Env. 35, 2001, 877-890) wird das Abgasfahnenmodell PLURIS beschrieben. Das Computerprogramm *IBJpluris* ist eine Modell-Implementierung von PLURIS und einem meteorologischen Präprozessor zur Erzeugung der Vertikalprofile von Windgeschwindigkeit, Windrichtung und Temperatur gemäß Richtlinie VDI 3783 Blatt 8 (2017).

Das Modell ist in verschiedenen Untersuchungen eingesetzt worden und wurde auch auf unabhängiger Basis nachimplementiert.[1]

[1] Presotto, L., Bellasio, R., Bianconi, R. (2005): *Assessment of the visibility impact of a plume emitted by a desulphuration plant.* Atmospheric Environment 39, 719-738.

Die aktuelle Version 3.1 implementiert darüber hinaus die Konventionen und Festlegungen, die in dem *Bericht zur Umweltphysik* Nr. 10 (*Vorschrift zur Berechnung der Abgasfahnenüberhöhung von Schornsteinen und Kühltürmen*, September 2017, siehe www.janicke) aufgeführt sind.

2 Das Programm

Das Modell ist als Computer-Programm in JAVA realisiert. Es ist ein reines Text-Programm, läuft also unter UNIX in einem Terminalfenster, unter Windows in einem DOS-Fenster.

2.1 Aufruf

Das Programm arbeitet nicht interaktiv, sondern verwendet die Parameterdatei `IBJpluris.txt`, in der alle Parameter für die Rechnung spezifiziert sind. Am Ende der Rechnung schreibt das Programm ein Arbeitsprotokoll in die Datei `IBJpluris.log`, den Verlauf der Fahnenparameter in die Datei `IBJpluris.dmna` und eine grafische Darstellung des Verlaufs in die Datei `IBJpluris.pdf`. Alle diese Dateien befinden sich in einem eigenen Verzeichnis, dem Arbeitsverzeichnis, das vom Benutzer angelegt werden muss.

Das Programm wird in Form einer JAR-Datei ausgeliefert, die mit JAVA gestartet wird. Ein *JAVA Runtime Environment* (JRE) kann kostenlos im Internet bezogen werden und entweder im Betriebssystem oder lokal installiert werden. PLURIS wurde mit der JAVA-Distribution Version 1.8 von Oracle erstellt und getestet.

Der Aufruf erfolgt mit JAVA in der Form (unter Windows mit entgegengesetztem Schrägstrich)

`java -jar jar/IBJpluris.jar` *Arbeitsverzeichnis Option ...*

oder, wenn z.B. ein lokales JRE im Unterordner `jre` installiert ist, mit

`jre/bin/java -jar jar/IBJpluris.jar` *Arbeitsverzeichnis Option ...*

Mögliche Optionen sind:

`-e`*Extension* : Den Namen der Ein- und Ausgabedateien wird *Extension* angehängt. Wird beispielsweise *Extension* auf `-1` gesetzt, dann wird die EIngabedatei `IBJpluris-1.txt` erwartet und die Ausgabe erfolgt in die Dateien `IBJpluris-1.log` und `IBJpluris-1.dmna`. Die *Extension* gilt ebenfalls für die Grafik- und die Profildatei.

`--ignore-profiles` : Eine eventuell vorhandene Datei mit den meteorologischen Profilen (standardmäßig Datei `profiles.dmna`) wird ignoriert (Standard: sie wird verwendet).

`--stop-at-finalrise` : Wenn ein Abbruchkriterium erfüllt ist, wird die Rechnung abgebrochen (Standard: Rechnung bis zur vorgegebenen Entfernung).

`--sim-profiles` : Es werden Ähnlichkeitsprofile zur Bestimmung des Flüssigwassergehaltes verwendet (Standard: Stufenprofil).

`--skip-graph` : Es wird keine grafische Ausgabe (Datei `IBJpluris.pdf`) erzeugt (Standard: Grafikausgabe).

`--skip-dmn` : Es wird keine Ergebnisausgabe (Datei `IBJpluris.dmna`) erzeugt (Standard: Ergebnisausgabe).

`--skip-stacktip-downwash` : Es wird kein *stacktip downwash* berücksichtigt.

`--skip-log` : Es wird keine Protokollausgabe (Datei `IBJpluris.log`) erzeugt (Standard: Protokollausgabe).

`--paint-extended` : In der grafischen Ausgabe werden zusätzlich das Ergebnis der bisherigen VDI-Richtlinie, die approximierten Verläufe im Lagrange'schen Partikelmodell und die Ränder der sichtbaren Fahne (a) dargestellt (Standard: Darstellung der Fahnenachse, des Fahnenrands $R/\sqrt{3}$ und der Abbruchkriterien).

`--`*Parameter*`=`*Wert* : Nach Einlesen der Eingabedatei wird der Wert des Parameters *Parameter* durch *Wert* überschrieben. Diese Option kann mehrmals verwendet werden.

`--break-factor=`*f* : Die Faktor für das Abbruchkriterium wird mit dem neuen Wert überschrieben.

Die Parameterdatei `IBJpluris.txt` ist eine reine Textdatei, kann also vom Benutzer mit einem beliebigen Text-Editor erstellt werden. Die Protokolldatei `IBJpluris.log` ist ebenfalls eine Textdatei, die vom Programm bei jedem Programmlauf angelegt wird. Die Rechenergebnisse werden in die DMN-Datei `IBJpluris.dmna` geschrieben. Sie enthält im Dateikopf die wesentlichen Eingabeparameter und die verwendeten meteorologischen Profile.

Die Parameterdatei besteht aus mehreren Abschnitten. Jeder Abschnitt definiert einen Parameterblock. Eine vollständige Auflistung aller möglicher Parameter ist in Abschnitt 2.2 gegeben.

Die erste Zeile eines Abschnittes beginnt mit dem Zeichen „*“, unmittelbar gefolgt von einem Namen, und definiert die Teilaufgabe, die ausgeführt werden soll, beispielsweise

```
*Quelle
```

Signifikant ist hierbei nur der erste Buchstabe, also das „Q“, wobei auch nicht zwischen Klein- und Großschreibung unterschieden wird. Genauso gut hätte man also auch schreiben können

```
*q
```

Es folgen in diesem Falle die Parameter, die die Quelle charakterisieren, also z.B.

```
*Quelle             ' Angaben zur Quelle
hq  100             ' Quellhöhe 100m
dq    5             ' Durchmesser 5m
tq   80             ' Austrittstemperatur 80 Grad Celsius
uq   17             ' Ausströmgeschwindigkeit 17m/s
```

Bei den Parameternamen wird nicht zwischen Klein- und Großschreibung unterschieden. Anschließend folgt, durch Leerzeichen und/oder Tabulator getrennt, der Wert. Dabei kann es sich um eine ganze Zahlen, eine Gleitkommazahl oder eine Zeichenkette handeln. Zeichenketten können in Hochkommata eingeschlossen sein.

Ein Apostroph in einer Zeile leitet einen Kommentar ein, der beim Einlesen überschlagen wird. Beginnt eine Zeile mit einem Minuszeichen oder einem Doppelkreuz, wird die ganze Zeile als Kommentar angesehen.

Die Namen der Parameter sind über alle Abschnitte hinweg eindeutig und dürfen nur in dem Abschnitt gesetzt werden, für den sie vereinbart sind.

2.2 Eingabeparameter

Das Programm *IBJpluris* erkennt folgende Abschnitte:

`*q` Quellparameter

`*a` Charakterisierung der Umgebung und zur Bestimmung der meteorologischen Profile

`*c` Rechenparameter

Die einzelnen Eingabeparameter sind diesen Abschnitten zugeordnet, dürfen also nur in dem betreffenden Abschnitt gesetzt werden. Sie stellen entweder ganze Zahlen, Gleitkommazahlen oder Zeichenketten dar und werden zu Beginn der Rechnung mit einem Standardwert belegt. Alle Längen sind in Metern anzugeben, alle Zeiten in Sekunden, alle Temperaturen in Grad Celsius.

In der folgenden Tabelle steht für jeden Parameter in einer Zeile der Name, anschließend der Datentyp (*integer*, *float* oder *string*), dahinter in Klammern die Anzahl der Werte und schließlich die Standardsetzung. In den darauf folgenden Zeilen ist die Bedeutung und Handhabung dieses Parameters beschrieben.

`*q` Quelle

`a1` *float*(1) `90`
Austrittswinkel gegen die Horizontale (Grad).

`a2` *float*(1) `0`
Austrittswinkel gegen die x-Achse gegen den Uhrzeigersinn (Grad).

`cq` *float*(1) `1`
Anfangskonzentration in der Fahne (ME/m^3).

`dq` *float*(1) `1`
Durchmesser der Quelle (m).

`fq` *float*(1)
Quadrat der Froudezahl beim Quellaustritt. Ist sie mit einem positiven Wert vorgegeben, so wird der Parameter `tq` ignoriert und die Austrittstemperatur berechnet.

`hq` *float*(1) `10`
Bauhöhe der Quelle (m).

`id` *string*(1)
Kurzinformation zur Rechnung.

`iq` *float*(1) `0.1`
Turbulenzintensität der Fahne bei Quellaustritt.

`lq` *float*(1) `0`
Spezifischer Flüssigwassergehalt bei Quellaustritt (Masse Flüssigwasser pro Masse feuchter Luft).

`qq` *float*(1) `-1`
Wärmestrom in MW (nach TA Luft) zur Berechnung der Austrittstemperatur. Es wird empfohlen, direkt die Austrittstemperatur `tq` vorzugeben!

`rq` *float*(1) `0`
Relative Feuchte bei Quellaustritt (Bruchteil der spezifischen Feuchte bei Sättigung).

`sq` *float*(1) `0`
Spezifische Feuchte bei Quellaustritt (Masse Wasserdampf pro Masse feuchter Luft). Überschreibt `rq`.

`tq` *float*(1) `10`
Austrittstemperatur (Grad Celsius).

`uq` *float*(1) `10`
Austrittsgeschwindigkeit (m/s).

`xq` *float*(1) `0`
X-Koordinate des Quellmittelpunktes (m).

`yq` *float*(1) `0`
Y-Koordinate des Quellmittelpunktes (m).

`zq` *float*(1) `0`
Wasserbeladung (Masse Gesamtwasser pro Masse trockener Luft) bei Quellaustritt. Überschreibt `sq`, `rq`, `lq`.

`*a` Umgebung

`ca` *float*(1) `1`
Konzentration in der Umgebungsluft (ME/m^3).

`d0` *float*(1) `3.0`
Verdrängungshöhe (m). Wird nur bei `lm`/`ki` ungleich 0 ausgewertet, siehe Abschnitt 2.3.

`da` *float*(1) `270`
Windrichtung gegen Nord im Uhrzeigersinn (Grad) in Meßhöhe. Dies ist die in der Meteorologie übliche Definition. Ein Wert von 270 Grad bedeutet Westwind parallel zur *x*-Achse. Die horizontale Austrittsrichtung der Fahne wird relativ zur *x*-Achse gegen den Uhrzeigersinn angegeben.

`ef` *float*(1) `1`
Faktor, mit dem die Einmischfunktion multipliziert wird.

`fc` *float*(1) `1.1e-4`
Coriolis-Parameter (1/s).

`ha` *float*(1) `10`
Anemometerhöhe (m). Wird nur bei `lm`/`ki` ungleich 0 ausgewertet, siehe Abschnitt 2.3.

`hm` *float*(1) `0`
Mischungsschichthöhe (m) für das meteorologische Grenzschichtmodell. Wird nur bei `lm`/`ki` ungleich 0 ausgewertet, siehe Abschnitt 2.3. Falls der Wert 0 ist, wird sie intern bestimmt.

`ia` *float*(1) `0.1`
Turbulenzintensität des Windfeldes in der Umgebung.

`ki` *integer*(1) `0`
Klassenindex (1: Klug/Manier I bis 6: Klug/Manier V), aus dem intern die Obukhov-Länge für das meteorologische Grenzschichtmodell (siehe Abschnitt 2.3)festgelegt wird. Ist `lm` ungleich 0, wird hieraus der Wert von `ki` bestimmt, unabhängig, was für `ki` vorgegeben ist.

`lm` *float*(1) `0`
Obukhov-Länge (m) für das meteorologische Grenzschichtmodell (siehe Abschnitt 2.3). Ist `lm` gleich 0 und `ki` ungleich 0, wird aus `ki` der klassierte Wert von `lm` bestimmt.

`nz` *float*(1) `300`
Anzahl der vertikalen Stützpunkte zur Festlegung der Profile.

`pa` *float*(1) `101300`
Standarddruck am untersten Stützpunkt (Pa). Das Vertikalprofil wird automatisch berechnet.

`ra` *float*(1) `0.0`
Relative Feuchte (Bruchteil der spezifischen Feuchte bei Sättigung) in Meßhöhe.

`ta` *float*(1) `10`
Temperatur in Meßhöhe (C).

`th` *float*(1) `2`
Meßhöhe der Temperatur und Feuchte (m). Wird nur bei `lm` gleich 0 ausgewertet, siehe Abschnitt 2.3.

`tx` *float*(1) `-0.008`
Temperaturgradient (C/m). Wird bei `lm`/`ki` gleich 0 direkt, sonst als adiabatischer Temperaturgradient verwendet, siehe Abschnitt 2.3.

`ua` *float*(1) `3`
Windgeschwindigkeit in Meßhöhe (m/s).

`uh` *float*(1) `10`
Meßhöhe der Windgeschwindigkeit (m). Wird nur bei `lm`/`ki` gleich 0 ausgewertet, siehe Abschnitt 2.3.

`us` *float*(1) `0`
Schubspannungsgeschwindigkeit (m/s). Wird nur bei `lm`/`ki` ungleich 0 ausgewertet, siehe Abschnitt 2.3. Ist der Wert 0, wird sie aus dem Windprofil berechnet.

`ux` *float*(1) `0`
Exponent für das Vertikalprofil der Windgeschwindigkeit. Wird nur bei `lm`/`ki` gleich 0 ausgewertet, siehe Abschnitt 2.3.

`vs` *float*(1) `0`
Sedimentationsgeschwindigkeit (m/s). Sedimentation wird als konstantes Absinken der Fahnenachse modelliert.

`z0` *float*(1) `0.5`
Rauhigkeitslänge (m). Wird nur bei `lm`/`ki` ungleich 0 ausgewertet, siehe Abschnitt 2.3.

`za` *float*(1) `10`
Vertikale Schrittweite zur Erzeugung der Profile (m).

`*c` Calculation

`oo` *integer*(1) 1
Ausgabemodus. Er gibt an, welche Fahnenparameter in die Ergebnisdatei ausgeschrieben werden. Folgende Werte sind möglich:

`0`: Ausgabe von (Längen skaliert mit `sc`)
`ll` (Distanz in der horizontalen Projektion),
`hh` (Höhe über Bauhöhe der Quelle),
`brg` (Überhöhung nach Briggs, kombinierte Fahne),
`brgm` (Überhöhung nach Briggs, Impulsfahne),
`brgb` (Überhöhung nach Briggs, thermische Fahne),
`vdid` (Überhöhung nach bisheriger VDI 3782 Blatt 3).

`1`: Ausgabe von
`xx` (x-Koordinate, m),
`yy` (y-Koordinate, m),
`zz` (z-Koordinate, m),
`w1` (Differenzgeschwindigkeit, m/s),
`hh` (Höhe über Bauhöhe der Quelle, m),
`ss` (Distanz entlang der Fahnenachse, m),
`ll` (Distanz in der horizontalen Projektion, m),
`zt` (Reisezeit, s),
`rr` (Radius, m),
`aa` (sichtbarer Radius bei kondensiertem Wasserdampf, m),
`uu` (Geschwindigkeit, m/s),
`tt` (Temperatur, C),
`cc` (Konzentration, ME/m^3).
`jj` (Geschwindigkeitsfluktuationen, m/s),
`qi` (spezifische Feuchte),
`ri` (relative Feuchte),
`li` (spezifischer Flüssigwassergehalt),
`dd` (Dichte, kg/m^3),
`ee` (Einmischfunktion, m^2/s).

`2`: Wie 1, aber Längen skaliert mit `sc`.

`3`: Ausgabe von
`xx` (x-Koordinate, m),
`yy` (y-Koordinate, m),
`zz` (z-Koordinate, m),
`w1` (Differenzgeschwindigkeit, m/s),
`hh` (Höhe über Bauhöhe der Quelle, m),
`ss` (Distanz entlang der Fahnenachse, m),
`ll` (Distanz in der horizontalen Projektion, m),
`zt` (Reisezeit, s),
`rr` (Radius, m),
`aa` (sichtbarer Radius bei kondensiertem Wasserdampf, m),
`ww` (horizontale Entfernung zur Quelle, $\sqrt{(\mathtt{xx}-\mathtt{xq})^2+(\mathtt{yy}-\mathtt{yq})^2}$, m),

`l1r` (Fahnenunterkante, `ll` + (`rr`/ $\sqrt{3}$) sin(`w1`), m),
`z1r` (Fahnenunterkante, `zz` − (`rr`/ $\sqrt{3}$) cos(`w1`), m),
`l2r` (Fahnenoberkante, `ll` − (`rr`/ $\sqrt{3}$) sin(`w1`), m),
`z2r` (Fahnenoberkante, `zz` + (`rr`/ $\sqrt{3}$) cos(`w1`), m),
`l1a` (sichtbare Fahnenunterkante, `ll` + `aa` sin(`w1`), m),
`z1a` (sichtbare Fahnenunterkante, `zz` − `aa` cos(`w1`), m),
`l2a` (sichtbare Fahnenoberkante, `ll` − `aa` sin(`w1`), m),
`z2a` (sichtbare Fahnenoberkante, `zz` + `aa` cos(`w1`), m).
`w1a` (sichtbare Fahnenunterkante, `ww` + `aa` sin(`w1`), m).
`w2a` (sichtbare Fahnenoberkante, `ww` − `aa` sin(`w1`), m).

`4`: Wie 3, aber Längen skaliert mit `sc`.

`sc` *float*(1) `1`
Skalierungslänge (m).

`sd` *float*(1) `0.01`
Mit `sc` skalierte Rechenschrittweite.

`se` *float*(1) `1000`
Mit `sc` skalierte Distanz auf der Fahnenachse, bis zu der gerechnet wird.

`sn` *integer*(1) `100`
Anzahl der Ausgabeschritte.

2.3 Meteorologische Profile

IBJpluris benötigt mindestens die Vertikalprofile der Windgeschwindigkeit, Windrichtung und Temperatur der Umgebungsluft. Zur Festlegung gibt es drei Möglichkeiten:

1. Parameter `lm` (Obukhov-Länge) und `ki` (Klassenindex) haben den Wert 0 (Standard): Die Windrichtung ist konstant und hat den Wert `da`. Die Windgeschwindigkeit u und Temperatur T werden als

$$u(z) = u_\mathrm{a}\left(\frac{z}{h_\mathrm{u}}\right)^m \quad (1)$$

$$T(z) = T_\mathrm{a} + \gamma(z - h_\mathrm{t}) \quad (2)$$

   mit u_a (`ua`), h_u (`uh`), m (`ux`), T_a (`ta`), γ (`tx`), h_t (`th`) angesetzt. Für $z > 200$ m ist u konstant.

2. Parameter `lm` oder `ki` ist ungleich 0: Die Profile von Windgeschwindigkeit und Temperatur werden nach Richtlinie VDI 3783 Blatt 8 (2017) bestimmt. Hierbei werden die Parameter `z0`, `d0`, `lm`, `ha`, `ta`, `da` und `ua` ausgewertet, zusätzlich kann `hm` und `us` vorgegeben werden.

3. Im Arbeitsverzeichnis existiert die Profildatei `profiles.dmna`: In diesem Fall werden die Profile von Windgeschwindigkeit, Windrichtung, Turbulenzintensität, Temperatur, relativer Feuchte und Umgebungskonzentration aus dieser Datei übernommen, es sei denn,

beim Programmaufruf wird die Option `-ignore-profiles` benutzt. Der Datenteil ist eindimensional und die Parameterwerte müssen für die verschiedenen Höhen in oben genannter Reihenfolge als *float*-Zahlen aufgeführt werden. Beispiel:

```
form   "ha%5.1f" "ua%5.1f" "da%5.1f" "ia%5.3f" "ta%5.1f" "ra%5.3f" "ca%12.2e"
mode  "text"
sequ  "i"
dims  1
size  28
lowb  1
hghb  12
*
   0.0   0.0 270.0 1.000  12.0 0.770   0.0
   5.0   2.0 270.0 1.000  12.0 0.770   0.0
  10.0   5.0 270.0 1.000  12.0 0.770   0.0
  20.0   6.5 270.0 1.000  12.0 0.770   0.0
  30.0   8.0 270.0 1.000  12.0 0.770   0.0
  40.0   9.0 270.0 1.000  12.0 0.770   0.0
  60.0  11.0 270.0 1.000  12.0 0.770   0.0
  80.0  12.5 270.0 1.000  12.0 0.770   0.0
 100.0  14.0 270.0 1.000  12.0 0.770   0.0
 200.0  19.0 270.0 1.000  12.0 0.770   0.0
 400.0  26.0 270.0 1.000  12.0 0.770   0.0
 600.0  30.0 270.0 1.000  12.0 0.770   0.0
-
```

Die verwendeten Profile werden in den Dateikopf der Ergebnisdatei ausgeschrieben.

A Dateistruktur von DMN-Dateien

Alle Dateien, die Ein- oder Ausgabefelder (Tabellen) repräsentieren, sind nach dem gleichen Prinzip aufgebaut. Sie enthalten zuerst einen Kopf, in dem alle Angaben zur Struktur und Darstellung der Tabelle stehen. Es folgt der Rumpf mit der eigentlichen Tabelle. Dieser Rumpf kann unmittelbar an den Kopf angehängt sein, wenn die Tabelle formatiert ausgeschrieben wird. Bei unformatierter Darstellung bildet der Rumpf immer eine eigene Datei. In dieser Binärdatei stehen die Tabellenelemente so, wie sie intern dargestellt sind, unmittelbar hintereinander ohne jegliche Steuerzeichen. Ein Tabellenelement kann eine einzelne Zahl (Datenelement) oder ein Verbund von mehreren Zahlen (*record*) sein.

Der Kopf ist eine Textdatei mit der Namenserweiterung `dmna`, die aus einzelnen Zeilen besteht. In jeder Zeile ist ein Parameter definiert. Der Name des Parameters steht zu Beginn der Zeile, gefolgt von einem oder mehreren Werten. Zulässige Trennungszeichen sind Leerzeichen, Tabulator und Semikolon, die einzeln oder kombiniert verwendet werden können. Die Zeile kann durch LF oder CR+LF abgeschlossen sein.

Neben den vom Programm benötigten Parametern kann der Kopf auch weitere Parameter enthalten. Parameter, die das Programm nicht kennt, werden ignoriert. Der Kopf endet mit einer Zeile, die zu Anfang einen Stern, also `*`, enthält. Mit der nächsten Zeile beginnt der Rumpf, sofern er mit dem Kopf zusammen eine einzige Datei bildet. Die (formatierten) Tabellenelemente im Rumpf werden durch Leerzeichen, Semikolon, Tabulator, CR oder LF getrennt. Die Tabelle wird durch eine Zeile, die mit drei Sternen beginnt, beendet.

Folgende Parameter im Kopf der Datei werden vom Programm erkannt und interpretiert (die Namen müssen klein geschrieben sein):

`data` *string*(1)
: Name der Datei, die die eigentliche Tabelle enthält. Ist `data` nicht spezifiziert oder hat es den Wert „`*`“, dann wird bei formatierter Ausgabe die Tabelle in die gleiche Datei geschrieben wie der Kopf. Bei unformatierter Ausgabe wird der Dateiname des Kopfes übernommen, er erhält aber statt „`.dmna`“ die Endung „`.dmnb`“. Falls in `data` eine Pfadangabe gemacht ist, gilt diese relativ zu dem Ordner, in dem der Kopf gespeichert ist.

`dims` *integer*(1)
: Anzahl der Dimensionen (maximal 5).

`fact` *float*(1)
: Faktor, mit dem bei formatierter Ausgabe alle Datenelemente vom Typ *float* oder *double* multipliziert werden, bevor sie im angegebenen Format ausgeschrieben werden. Bei der Eingabe formatierter Daten werden diese Datenelemente nach dem Einlesen durch `fact` dividiert. Der Faktor wirkt nur auf die Datenelemente, für die in der Formatangabe (siehe `form`) kein eigener Faktor angegeben ist.

`form` *string*(1)
: Format, nach welchem bei formatierter Speicherung die Daten abgelegt sind. Bestehen die Tabellenelemente des gespeicherten Feldes aus mehreren Datenelementen, dann ist für jedes Datenelement eine Formatangabe erforderlich, und alle Einzelformate verkettet ergeben die Zeichenkette `form`.
$Format = Format_1 Format_2 ...$

$Format_i$ = *Name*`%(*`*Factor*`)`*Length*`.`*PrecisionSpecifier*

Es bedeuten:

Name	Name des Datenelementes (optional).
Factor	Skalierungsfaktor (optional einschl. Klammern).
Length	Länge des Datenfeldes.
Precision	Anzahl der Nachkommastellen (bei *float*-Zahlen).
Specifier	Umwandlungsangabe.

Der Skalierungsfaktor *Factor* wird genauso gehandhabt wie der Parameter `fact`. Die Längenangabe *Length* ist die Mindestlänge des Datenfeldes. Sie kann überschritten werden, wenn dies zur korrekten Darstellung der Zahl erforderlich ist. Zwischen den Zahlen steht immer mindestens ein Trennungszeichen.

Folgende Umwandlungsangaben sind möglich:

Spec.	Typ	Länge	Beschreibung
`c`	*character*	1	einzelne Buchstaben
`d`	*integer*	4	Dezimalzahl
`x`	*integer*	4	Hexadezimalzahl
`f`	*float*	4	Festkommazahl (ohne Exponent)
`e`	*float*	4	Gleitkommazahl (mit Exponent)
`t`	*integer*	4	Zeitangabe (ohne Datum)

Den Angaben `f` und `e` kann ein `l` vorangestellt sein (*double* mit Länge 8 Bytes), den Angaben `d` und `x` ein `h` (*short integer* mit der Länge 2 Bytes).

Zeitdarstellung bei Binärausgabe: Bei Zeitangabe ohne Datum bezeichnet die Zahl die vergangenen Sekunden. Ist der Angabe `t` ein `l` vorangestellt, wird die Zahl (*double* mit Länge 8 Bytes) als Zeitangabe mit Datum interpretiert: Die Vorkommastellen bezeichnen die Anzahl der Tage seit 1899-12-30.00:00:00 plus 10^6, die Nachkommastellen den Anteil der vergangenen Sekunden an diesem Tag. Zeitdarstellung bei Textausgabe: Bei der Angabe `t` hat die Zeitangabe die Form `dd.hh:mm:ss` oder `hh:mm:ss`, bei der Angabe `lt` die Form `yyyy-mm-dd.hh:mm:ss`.

Gleichartige Formatangaben können zusammengefaßt werden:
`vx%5.2fvy%5.2fvz%5.2f` ist äquivalent zu `vx%[3]5.2f`[2]

`hghb` *integer*(`dims`)
Höchster Indexwert für die verschiedenen Laufindizes.

`lowb` *integer*(`dims`)
Niedrigster Indexwert für die verschiedenen Laufindizes.

`mode` *string*(1)
Bei `binary` sind die Daten unformatiert gespeichert, sonst formatiert.

`sequ` *string*(1)
Angabe, in welcher Indexfolge die Daten gespeichert sind. Normalerweise läuft der am

[2] Bei zusammengefaßten Formatangaben gilt der angebene *Name* nur für das erste Element. Bei den folgenden Elementen wird der letzte Buchstabe in *Name* jeweils um eine Stelle im Alphabet weitergerückt, so daß schließlich der angegebene Ausdruck entsteht.

weitesten rechts stehende Index am schnellsten (C-Konvention). Dies entspricht bei einem 3-dim. Feld A_{ijk} der Angabe `i+,j+,k+`. FORTRAN speichert gemäß `k+,j+,i+`. Ein Minuszeichen statt des Pluszeichens bedeutet, daß der betreffende Index rückwärts läuft. Es können auch Teilbereiche ausgewählt werden:
`j=10..1/1,i=5..25/1,k=1`. Die Angabe */n* bedeutet, daß der betreffende Index des Ausschnittes mit dem Wert *n* anfängt. Wird mit `sequ` ein Auschnitt des Datenfeldes definiert, dann beziehen sich die Indexgrenzen `lowb` und `hghb` auf die ursprünglichen Indexdefinitionen.

`size` *integer*(1)
Länge der einzelnen Daten (*record size*) in Bytes. Bei formatierter Speicherung muß die aus der Formatangabe resultierende Summe der Längen der einzelnen Datenelemente gleich `size` sein.

Zeichenketten müssen, wenn sie Leerzeichen enthalten, in Doppelhochkomma eingeschlossen sein, ansonsten sind diese optional.

Beispiel:

Ein Feld von Gleitkommazahlen $A_{ijk} = 100i + 10j + k, i = 1..3, j = 2..4, k = 0..1$ wird in horizontalen Schichten gespeichert:

```
form  "%4.1f"
mode  "text"
sequ  "k+,j-,i+"
fact  1.000e-001
dims  3
size  4
lowb  1 2 0
hghb  3 4 1
*
  14.0  24.0  34.0
  13.0  23.0  33.0
  12.0  22.0  32.0

  14.1  24.1  34.1
  13.1  23.1  33.1
  12.1  22.1  32.1

***
```