Forschungszentrum Karlsruhe
Technik und Umwelt

Wissenschaftliche Berichte

FZKA 5995

# Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen

## Teil 1: Einführung und Überblick

Siegfried Kleinheins

Institut für Neutronenphysik und Reaktortechnik
Projekt Nukleare Sicherheitsforschung

Forschungszentrum Karlsruhe GmbH, Karlsruhe
1997

**Zusammenfassung**

VISART ist ein standardisiertes Format für Postprocessor-Dateien mit Daten, wie sie von Fluiddynamik- und ähnlichen Codes über einer mehrdimensionalen Geometrie und der Zeit errechnet werden. VISART-Dateien sind in erster Linie für die grafische Auswertung der Ergebnisse in einem separaten Lauf gedacht, enthalten aber keine Steuerinformation für die Grafik. Das VISART-Konzept einer einheitlichen Schnittstelle zwischen Codes und Auswerteprogrammen erlaubt einer Vielzahl von Codes, eine Vielzahl von Diensten zur Weiterbehandlung der berechneten Daten zu nutzen, und zwar mit minimalem Implementierungsaufwand. Dieser Bericht führt in das VISART-Konzept ein, stellt die geometrischen und anderen Annahmen zusammen, für die der VISART-Standard entworfen wurde, gibt einen Überblick über den Aufbau und das Format von VISART-Dateien, und stellt anhand einer grafischen Benutzeroberfläche (unter UNIX) die VISART-Dienste zur Inspektion und Manipulation von VISART-Dateien und zur Visualisierung ihres Inhalts vor.

# The VISART Concept of a Standardized Interface between Codes and Evaluation Programs

## Part 1: Introduction and Survey

**Abstract**

VISART is a standardized format for postprocessor files with data calculated by fluid-dynamics and similar codes over a multi-dimensional geometry and time. VISART files primarily provide for the graphical evaluation of the results in a separate run, but they do not include graphics control information. The VISART concept of a uniform interface between codes and evaluation programs allows the use of a variety of services for a diversity of codes, with minimal implementation effort. This report introduces the VISART concept, states the geometrical and other assumptions for which the VISART standard was developped, sketches the structure and the format of VISART files, and in the light of a graphical user interface (under UNIX) presents the VISART services for the inspection and manipulation of VISART files and for the visualization of their contents.

# Übersicht über das VISART-Konzept und seine Dokumentation

### Welche Aufgabe ist zu lösen ?

Die rechnerische Simulation von Problemen der Fluiddynamik, Elektrodynamik, Neutronik usw. reduziert sich meist auf die numerische Lösung von zeitabhängigen partiellen Differentialgleichungen mittels räumlicher und zeitlicher Diskretisierung. Die dafür verwendeten Computer-Codes errechnen dabei eine Fülle von Daten über der Geometrie und der Zeit. Zu ihrer Auswertung sind diese Daten in geeigneter Form auf einer Postprocessor-Datei bereitzustellen.

### Was ist VISART ?

VISART ist in erster Linie der Name eines (zunächst im FZK-Bereich) standardisierten Formats für Postprocessor-Dateien mit nach obiger Art errechneten Daten, die zur Auswertung erstellt werden, aber noch keine Steuerinformation für Grafik usw. enthalten. Im weiteren Sinne ist VISART der Name eines auf der Standardisierung aufbauenden Konzepts einer einheitlichen Schnittstelle zwischen Codes und Auswerteprogrammen, die es erlaubt, mit minimalem Implementierungsaufwand einer Vielzahl von dafür geeigneten Codes eine Vielzahl von Diensten zur Weiterbehandlung der berechneten Daten zur Verfügung zu stellen.

### Welche Codes erstellen VISART-Dateien ?

Postprocessor-Ausgabe im VISART-Format wurde (Stand Juni 1997) bereits in die folgenden, im FZK betreuten oder entwickelten Codes eingebaut: SIMMER-II.12, AFDM, SIMMER-III, HYDSOL, IVA-KA, FLUTAN, KADI2D, MAX3D, BFCPIC, GASFLOW.

### Was kann man mit VISART-Dateien anfangen ?

VISART-Dateien können zunächst einmal mit einem Dienstprogramm in wählbarem Umfang und Detail „durchblättert“ werden. Mit einem anderen Programm können die Werte ausgewählter Größen gezielt über der Geometrie und/oder der Zeit in Zahlenform „angeschaut“ werden.

Die graphische und anderweitige Auswertung von Größen einer VISART-Datei ist sodann mit einigen im Hause oder im Auftrag entwickelten Auswerteprogrammen möglich, die das VISART-Dateiformat direkt lesen können.

Die Visualisierung der Daten einer VISART-Datei mit kommerziellen Visualisierungssystemen (bis jetzt AVS, UNIGRAPH, Tecplot), die jeweils eigene Formate für ihre Eingabedateien haben, geschieht über das Zwischenschalten eines für VISART angefertigten Adapterprogramms, das die Eingabedatei des jeweiligen Systems für die gerade gewählten Größen und Darstellungskoordinaten (z.B. über der Geometrie—auch Schnitten oder Ausschnitten daraus—und/oder der Zeit) erstellt.

Schließlich steht ein mächtiges Werkzeug zum Bearbeiten von VISART-Dateien und zum Ableiten neuer Größen aus den in der VISART-Datei enthaltenen Daten zur Verfügung.

Alle erwähnten Dienste können über eine zentrale graphische Oberfläche auf X-Terminals für Workstations aufgerufen und gesteuert werden.

**Wo findet man was ?**

Teil 1 der VISART-Dokumentation führt in das VISART-Konzept ein und stellt die graphische Benutzeroberfläche zum Aufruf der Dienste vor. Das Programm VISARTUL zum Durchblättern und Prüfen von VISART-Dateien wird beschrieben, und es wird ein Überblick über die anderen zur Verfügung stehenden Dienste gegeben. (Dieser Teil ist die Grundlage für die nachfolgenden Teile.)

Teil 2 der VISART-Dokumentation definiert den VISART-Standard für den Aufbau der Postprocessor-Dateien.

Teil 3 der VISART-Dokumentation beschreibt das Adapterprogramm VISAPTER zum Erstellen von Eingabedateien für die Visualisierung mit AVS, UNIGRAPH, Tecplot usw. und zum Anschauen ausgewählter Werte in Zahlenform.

Teil 4 der VISART-Dokumentation beschreibt das Werkzeug VISARTOP zum Bearbeiten von VISART-Dateien und zum Ableiten neuer Größen.

Teil 5 der VISART-Dokumentation beschreibt das Adapterprogramm VISARVOL zum Erstellen von Eingabedateien für die zweidimensionale Darstellung von Volumenanteilen mit Tecplot usw..

Auswerteprogramme, die VISART-Dateien direkt lesen und verarbeiten können, werden nicht in dieser Reihe, sondern eigenständig dokumentiert. Das gleiche gilt für den Einbau der VISART-Schnittstelle in die einzelnen Codes.

## Vorwort

Auf Initiative des Institutsleiters des INR, Prof. Dr. G. Keßler, befaßte sich im Frühjahr 1991 ein Kreis von Autoren und Benutzern von Grafik-Programmen mit der zukünftigen „Visualisierung von Großprogrammen“. Konkreter Anlaß war die bevorstehende Beschaffung einer Workstation für diese Aufgabe (einer STARDENT-Workstation für das AVS-Visualisierungssystem) und die damit verbundene Umstellung vorhandener Plot-Software. Darüber hinaus sollte für die Zukunft eine gewisse Vereinheitlichung der Programme für grafische Darstellungen der Ausgabe von Großprogrammen erreicht werden.

Der Autor dieses Berichts wurde beauftragt, ein Standardformat für Postprocessor-Dateien von Fluiddynamik-Codes zu entwerfen. Solche Dateien dienen als Schnittstelle zwischen Anwendungs-Codes und Visualisierungsprogrammen im Wege der hierbei üblichen „off-line“ Visualisierung. Eine Standardisierung ist die Voraussetzung dafür, daß unterschiedliche Codes die gleichen Visualisierungsprogramme auf effiziente Weise benützen können. Da auch immer mehr Visualisierungsoptionen für die Code-Ausgaben zur Verfügung stehen, vervielfältigt sich das Einsparungspotential bei einer Standardisierung entsprechend.

Im Verlauf des Jahres 1991 wurde, nach Diskussion des Entwurfs mit Benutzern und nach erstmaligen Anwendungen, das Release 1.01 eines Standardformats, dem der Titel **VISART**[1] verliehen wurde, festgelegt und dokumentiert (1. Fassung von [1b]).

Seitdem wurde eine ganze Anzahl von Codes auf die Postprocessor-Ausgabe im VISART-Format umgestellt (siehe „Übersicht“), angefangen mit zweidimensionalen Codes, wie die der SIMMER-Linie [2a], [2b], [2c], bis zu dreidimensionalen Codes wie IVA3 [2e] bzw. IVA-KA und FLUTAN [2f]. Andererseits wurden einige im Hause erstellte Grafik- und Auswerteprogramme auf das VISART-Format umgestellt, aber vor allem die gängigen kommerziellen Visualisierungssysteme AVS, UNIGRAPH und Tecplot über ein Adapter-Programm für das VISART-Format zugänglich gemacht (1. Fassung von [1c], mit Ergänzungen).

Neben den Visualisierungsprogrammen wurden einige weitere Dienste für VISART-Dateien bereitgestellt, wie Programme zur Inspektion der Dateien in wählbarem Umfang und Detail und zur Manipulation des Dateiformats und Dateiinhalts (1. Fassung von [1d], mit Ergänzungen).

Besonders seit der Bereitstellung einer graphischen Benutzeroberfläche, zunächst für den IBM-Großrechner unter dem Betriebssystem MVS (1994), dann auch für Workstations unter UNIX (Anfang 1996), erfreute sich das VISART-Konzept einer wachsenden Zahl von Anwendern, die die Entlastung von der Grafikprogrammierung, die Flexibilität bei der Wahl der Visualisierungssysteme und -arten und den leichten Einstieg in die Visualisierung schätzen. Die Wechselwirkung mit den Anwendern hat zu vielen Verbesserungen und Erweiterungen der für die VISART-Dateien bereitgestellten Dienste geführt. Das ursprünglich definierte Release 1.0 des Standards mußte dabei nur geringfügig geändert werden, wurde aber um etliche Möglichkeiten (rückwärtskompatibel) erweitert.

Die inzwischen zur Verfügung stehende Palette von Diensten für das VISART-Konzept, die Neuerungen und Erweiterungen von Standard und Programmen, und der Umstieg von MVS auf UNIX mit dem entsprechenden Wechsel der Benutzeroberfläche machten eine Neufassung der Dokumentation überfällig. Die neugefaßten Berichte wurden in eine Reihe eingeordnet, denen nun der vorliegende Bericht als Einführung in das VISART-Konzept vorangestellt wird.

---

[1] von „**Vis**ualisierungsstand**art**“ (leider nur mit nicht standardgemäßer Schreibweise!) oder „**Vis**ualisierung **a**uf ***r*** und ***t***“ oder „The **Art** of **Vis**ualization“; mit dem „**Wizard** of Oz“ hat VISART nur die Farbe der Benutzeroberfläche gemein.

# Inhalt

# 1. Das VISART-Konzept

In diesem Kapitel werden das Konzept und die Vorteile der Datenvermittlung zwischen Computer-Codes und Auswerteprogrammen über eine einheitliche, standardisierte Schnittstelle vorgestellt.

## 1.1 Postprocessing allgemein

### 1.1.1 Postprocessing zur Auswertung von Daten aus Computer-Codes

Computer-Codes, speziell solche der in 1.1.2 beschriebenen Klasse, liefern als Ergebnis einer Rechnung eine Fülle von Daten, die in Form von ausgedruckten Tabellen nicht mehr zu überblicken wären. Seit die Möglichkeiten zur Verfügung standen, setzte man bei der Auswertung daher auf die grafische Darstellung mittels spezieller Hard- und Software. Heute hat sich die Computer-Grafik zu umfangreichen Visualisierungssystemen (siehe 1.1.3) fortentwickelt, die an Leistung und Komfort kaum noch Wünsche offen lassen.

Die Visualisierung[1] der berechneten Daten mit einem Visualisierungssystem läßt sich auf zweierlei Arten implementieren:

- Im ersten Fall sind im Code bereits Routinen aus der Bibliothek eines Visualisierungssystems eingebaut, die das Aussehen usw. der Bilder festlegen. Die grafische Ausgabe erscheint auf dem Bildschirm parallel zu der Berechnung im Code. Alternativ oder zusätzlich dazu wird die grafische Ausgabe auf speziellen Dateien zur späteren Anzeige gespeichert.

  Diese—„on-line“ oder „real-time“—Visualisierung hat den Nachteil, daß Umfang und Aussehen der grafischen Ausgabe bereits im Code festgelegt sind und nachträglich nicht mehr verändert werden können. Falls nicht ein standardisiertes Dateiformat verwendet wird, ist man zudem auf das einmal gewählte Visualisierungssystem festgelegt.

- Im zweiten Fall werden die Daten, die für eine Visualisierung in Frage kommen, zu bestimmten Problemzeitpunkten in geeigneter Form auf eine sog. Postprocessor-Datei geschrieben. Diese Datei enthält keine Information über die Grafik selbst. In einem vom Code getrennten Lauf wird diese Datei dann vom Visualisierungssystem gelesen bzw. dafür aufbereitet, und hier dann auch das Aussehen der Bilder festgelegt.

  Dieses Postprocessing—auch „off-line“ Visualisierung genannt—hat den Vorteil, daß Umfang und Aussehen der grafischen Ausgabe erst bei der Visualisierung selbst bestimmt werden und beliebig oft variiert werden können. Da unter dem UNIX-Betriebssystem auf die Postprocessor-Datei schon während ihrer Erstellung zugegriffen werden kann, können auch bei dieser Methode Bilder schon parallel zur Berechnung der Daten im Code angezeigt werden.

Der zweite Weg scheint sich in der Praxis durchzusetzen. Leider fehlte bisher ein Standard für das Format von Postprocessor-Dateien, wie es solche Standards für Dateien mit Grafik gibt. Gäbe es einen solchen Standard, so wäre man auch bei der Auswahl der Visualisierungssysteme völlig flexibel, und könnte erheblich an Implementierungsaufwand einsparen.

In dieser Arbeit wird ein im FZK entworfenes und eingesetztes Standardformat für Postprocessor-Dateien mit dem Namen **VISART** vorgestellt, und es werden die für so standardisierten Dateien bereitgestellten Dienste (Programme, Adapter, Benutzeroberflächen) einführend beschrieben.

[1] Unter „Visualisierung“ wollen wir hier auch konventionelle grafische Darstellung und andere Formen der Auswertung und Nachverarbeitung von Daten verstehen.

### 1.1.2 Einige Computer-Codes

Die Computer-Codes, auf die wir uns im folgenden beziehen, sind Codes zur rechnerischen Simulation von Problemen der Fluiddynamik, Elektrodynamik, Neutronik usw.. Sie lösen numerisch die zeitabhängigen partiellen Differentialgleichungen, auf die die physikalischen Probleme reduziert wurden, mittels räumlicher und zeitlicher Diskretisierung der Geometrie bzw. des interessierenden Zeitbereichs des Problems. Als Ergebnis erhält man die Werte physikalischer Größen (wie Druck, Dichte, Temperatur, Geschwindigkeit, usw.) an Punkten oder Maschen des geometrischen Netzes für die Problemzeitschritte.

In der folgenden Tabelle werden die Charakteristiken einiger solcher im FZK entwickelten oder betreuten Computer-Codes zusammengestellt. Alle Codes erzeugen Postprocessor-Ausgabe zur anschließenden Visualisierung der Rechenergebnisse (und zwar bereits im VISART-Format—siehe Bild 1).

| Code | Geometrische Dimension | Netz | Koordinaten-system | Kompaktheit des Netzes |
|---|---|---|---|---|
| **SIMMER-II.12** [2a] | 2 | regulär | $x, y$<br>$r, z$ | voll |
| **AFDM** [2b] | 2 | regulär | $r, z$ | voll |
| **SIMMER-III** [2c] | 2 | regulär | $x, y$<br>$r, z$ | voll |
| **HYDSOL** [2d] | 2 | regulär | $x, y$<br>$r, z$<br>$r, \varphi$<br>$r, \psi$ | voll |
| **IVA3** und **IVA-KA** [2e] | 3 | regulär | $x, y, z$<br>$\rho, \vartheta, z$ | voll |
| **FLUTAN** [2f] | 3 | regulär | $x, y, z$<br>$\rho, \vartheta, z$ | defektiv |
| **KADI2D** [2g] | 2 | irregulär | $x, y$<br>$r, z$ | voll |
| **BFCPIC** [2h] | 2 | irregulär | $x, y$<br>$r, z$ | Löcher |
| **GASFLOW** [2i] | 3 | regulär | $x, y, z$<br>$\rho, \vartheta, z$ | voll |

Erklärungen:

Reguläre und irreguläre Netze werden in 2.1.2 definiert; volle Netze, Löcher in Netzen und defektive Netze in 2.1.3.

### 1.1.3 Einige Visualisierungssysteme

Während früher zur grafischen Auswertung der berechneten Daten haus- oder benutzereigene, evtl. sogar code-spezifische, Grafikprogramme angefertigt werden mußten, werden heute für PCs und Workstations interaktive Visualisierungssysteme kommerzieller Anbieter eingesetzt. Diese sind speziell auf die Auswertung der Ergebnisse von Codes der oben beschriebenen Art zugeschnitten. Jedes neue Produkt bzw. Release übertrifft die älteren in Umfang, Leistung und Komfort. Dennoch hat jedes System seine eigenen Vor- und Nachteile und ist auf bestimmte Zwecke spezialisiert, z.B. im Hinblick auf

- Verarbeitungsart (batch oder interaktiv);
- Visualisierungsziel (Präsentation in Vorträgen, Grafik in Veröffentlichungen, wissenschaftliche Auswertung);
- Betriebssystem, verwendete grafische Basissoftware und/oder Anwendungssoftware.

Deshalb werden wohl auch in Zukunft mehrere Visualisierungssysteme nebeneinander benützt werden müssen.

Alle Systeme stellen die im verlangten Format eingegebenen Zahlenwerte der berechneten Größen ein-, zwei- oder evtl. auch dreidimensional über den ausgewählten Koordinaten dar, sei es als einfache Funktionsgrafik, als Höhenlinien oder Konturplots, perspektivische Plots, Vektorplots, oder auf raffiniertere Weise (z.B. in drei Dimensionen).

In der folgenden Tabelle sind die Fähigkeiten einiger kommerzieller Visualisierungssysteme zusammengestellt, die für die Visualisierung der Ergebnisse der oben genannten Computer-Codes im FZK zur Zeit in Gebrauch sind. Alle laufen auf Workstations (und sind bereits für VISART-Dateien zugänglich—siehe Bild 1).

| System | eff. Dim. | Reg. Netz kartesisch | Reg. Netz krummlinig | Irreg. Netz | Teilnetze | Leere Maschen (Löcher) | Defektive Netze | Cell-based Darst. | Festes Dateiformat |
|---|---|---|---|---|---|---|---|---|---|
| **AVS-FD**[1] Release 5 [3a] | 3 | x[(5)] | x[2] | x | | | | | x |
| **AVS-UCD**[1] Release 5 [3a] | 3 | x | x[2] | x | x[3] | x[3] | x | x | x |
| **UNIGRAPH** 2000 [3c] | 2 | x[5] | | | | x | x[4] | | |
| **Tecplot** Version 6 [3d] | 3 | x | x[2] | x | x | x | x[4] | x | x |

(Anmerkungen und Erklärungen siehe nächste Seite)

[1] FD = Field Data; UCD = Unstructured Cell Data
[2] durch Umrechnung auf kartesische Koordinaten eines irregulären Netzes
[3] durch explizites Aufzählen jeder darzustellenden Masche
[4] durch Abbildung auf Netze mit leeren Maschen (Löchern)
[5] Koordinatenübergabe in Form von Koordinaten pro Maschenindex
[(5)] Koordinatenübergabe in Form von Koordinaten pro Maschenindex oder in Form von Koordinatenpaaren/tripel pro Masche

Erklärungen:

Die effektive Dimension 3 gibt an, daß das System auch Darstellungsmöglichkeiten über 3D-Netzen hat, wie z.B. Isoflächen oder Vektorpfeile im Raum.

Netze und Koordinaten werden in 2.1.2 definiert, defektive Netze und leere Maschen („Löcher“) in 2.1.3, Teilnetze in 2.1.4.

Bei der Option der „cell-based Darstellung“ eines Netzes weist das System den Wert einer Größe der zugehörigen Masche als ganzer zu, anstatt nur, wie bei der „node-based Darstellung“ (die der Normalfall ist), einer Lokation, z.B. in der Maschenmitte. Dabei wird auch die Netz- und Maschengeometrie vom System automatisch richtig dargestellt, während bei der node-based Darstellung in der Regel eine halbe Maschenbreite an der Netzhülle fehlt, falls nicht den berechneten Werten in den Maschenmitten die Randwerte auf der Netzhülle überlagert werden:

„Festes Dateiformat“ bezieht sich auf das Format, in dem das System die zu visualisierenden Daten erwartet (siehe [1c]). Die nicht in dieser Spalte aufgeführten Systeme können die Daten auch lesen, sofern sie lediglich in einer tabellarischen Form angeliefert werden.

Alle diese Systeme (außer AVS-UCD) bieten darüber hinaus die Möglichkeit zur Erzeugung konventioneller Funktionsgrafik, wofür aber auch einfachere Grafiksysteme eingesetzt werden können.

### 1.1.4 Postprocessing traditionell

Die übliche Weise, Rechenergebnisse über Postprocessor-Dateien zu visualisieren, kann man etwa wie folgt charakterisieren.

- Sie legen sich für Ihren Code auf ein bestimmtes Grafik- oder Visualisierungssystem fest.
- Sie implementieren die Postprocessor-Ausgabe im von diesem System verlangten Format.
- Dabei dürfen Sie nicht vergessen,
  - die Netzgeometrie mit Rändern, Löchern, Teilnetzen usw. richtig darzustellen;
  - die Größen auf die richtigen Lokationen im Netz zu plazieren;
  - ggf. die Koordinaten umzurechnen;
  - für die Darstellung von Skalaren, aber auch von Vektoren zu sorgen;
  - Darstellungsmöglichkeiten über der Geometrie, über der Zeit und über Kombinationen von beiden Dimensionen einzuplanen;
  - an die Achsen- und Bildbeschriftung mit Problemtitel, Größennamen, Zeitpunkt, usw. zu denken;
  - die Behandlung von Restart-Dateien vorzusehen.
  - ...
- Sie lassen nun Ihren Code laufen und erzeugen Ihre Postprocessor-Dateien.
- Nach einiger Zeit entdecken Sie,
  - daß Sie auf ein anderes Grafik- oder Visualisierungssystem umsteigen wollen oder müssen, das ein ganz anderes Dateiformat verlangt;
  - daß Sie dazu mit Ihren Dateien auf Maschinen mit anderen Zahlendarstellungen gehen müssen, was in der Regel nur für formatierte Dateien möglich ist;
  - daß Sie Darstellungen von abgeleiteten Größen benötigen, die nicht direkt auf Ihrer Datei stehen;
  - daß Sie die Übersicht über den Inhalt Ihrer Postprocessor-Dateien verloren haben, weil Sie im Dateiformat keinen oder nicht genug Platz für die Dokumentation des Inhalts vorgesehen hatten.
- Wenn (nicht „falls“ !) einer dieser Fälle auftritt
  - müssen Sie eine Konversionsroutine für Ihre Postprocessor-Dateien schreiben oder einen anderen Trick anwenden;
  - oder Sie fangen wieder mit dem ersten Schritt dieser Liste an ...

Die „Effizienz“ dieser Vorgehensweise ist offensichtlich. Mit jedem Code, der auf diese Weise bearbeitet wird, vergrößert sich der Implementierungsaufwand entsprechend.

## 1.2 Postprocessing über die VISART-Schnittstelle

### 1.2.1 Postprocessing mit VISART-Dateien

Unser Ziel ist das Verfügbarmachen vorhandener oder neuer Visualisierungssysteme für alle in Frage kommenden Anwendungscodes mit möglichst geringem Implementierungsaufwand. Notwendige Voraussetzung dafür ist die Standardisierung der Postprocessor-Dateien als **Schnittstelle** zwischen Codes und Visualisierungssystemen. Dieser Standard muß genügend flexibel und portabel sein, um weitgehend angewandt werden zu können.

Die Forderung nach **Flexibilität** wird erfüllt, wenn durch das angestrebte standardisierte Dateiformat nur die Form, nicht der Inhalt (Umfang, Auswahl, Art, Bedeutung usw.), der zwischen Code und Visualisierungssystem vermittelten Information festgelegt wird. Es ist Sache des Codes und seiner Steuerung, die auf die Postprocessor-Datei auszugebenden Größen zu definieren; es ist Sache des Visualisierungssystems und seiner Steuerung, unter den auf der Postprocessor-Datei angebotenen Größen auszuwählen. Die Größen in der Postprocessor-Datei sollen im wesentlichen nur durch ihre Namen identifiziert werden können. Außer den Kenndaten und Werten der Größen darf die Datei keine weitere Information, insbesondere keine Steuerungsinformation für die Visualisierung, enthalten.

Die Forderung nach **Portabilität** ist erfüllt, wenn der Standard nur auf die Konventionen für die in jeder Programmiersprache vorhandenen Input/Output-Statements Bezug nimmt und neben unformatierten (oder binären) Dateien auch formatierte (oder ASCII) Dateien vorsieht, die von der maschineninternen Zeichendarstellung unabhängig sind.

Diese Forderungen werden durch den **VISART-Standard** für Postprocessor-Dateien erfüllt.

Dem in 1.1.4 charakterisierten Vorgehen können wir, bei Standardisierung der Postprocessor-Dateien, das folgende, wesentlich effizientere, gegenüberstellen.

- Implementieren Sie die Postprocessor-Ausgabe im VISART-Format in Ihren Code. (Dies ist eine Sache von wenigen Stunden bis Tagen, je nach der Übersichtlichkeit der Code-Struktur. Hierfür können Sie die Muster-Programmteile aus Anhang B.1 benutzen oder sich an Codes orientieren, die die VISART-Schnittstelle bereits implementiert haben.)
- Lassen Sie nun Ihren Code laufen und erzeugen Sie Ihre Postprocessor-Dateien.
- Holen Sie sich die VISART-Dienstprogramme und die grafische Benutzeroberfläche dazu auf Ihre Workstation, falls sie dort noch nicht installiert sind.
- Alle über die grafische Benutzeroberfläche angebotenen Dienste stehen Ihnen nun zur Verfügung, von der Visualisierung mit den angeschlossenen Systemen bis zur Dateimanipulation und der Erzeugung neuer Größen. Auf was Sie bei der traditionellen Implementierung der Visualisierung selbst zu achten hätten, wird Ihnen hier abgenommen, weil es im Standard und in den VISART-Dienstprogrammen bereits vorgesehen ist.

Neue Visualisierungssysteme oder neue Releases derselben werden bei Bedarf vom VISART-Systemverwalter über die Adapterprogramme und die Benutzeroberfläche angeschlossen und stehen damit sofort allen Anwendern von Codes mit VISART-Ausgabe zur Verfügung.

### 1.2.2 Datenfluß über die VISART-Schnittstelle

Im VISART-Konzept vermitteln die VISART-Postprocessor-Dateien den Datenfluß zwischen Codes und Visualisierungsprogrammen und wirken somit als einheitliche standardisierte Schnittstelle.

Bild 1 zeigt den Datenfluß über die VISART-Schnittstelle. Auf der linken Seite des Bildes sind einige der Codes aufgeführt, die VISART-Dateien erzeugen.

Auf der rechten Seite des Bildes sind einige Visualisierungs- und andere Auswerteprogramme aufgeführt, mit einer symbolischen Darstellung ihrer Fähigkeiten. Einige dieser Programme können VISART-Dateien direkt lesen und verarbeiten. Vor die kommerziellen Visualisierungssysteme ist dagegen ein Adapter-Programm (VISAPTER oder VISARVOL) geschaltet, das die VISART-Dateien in die vom jeweiligen Programm benötigte Eingabe umsetzt. Das Adapter-Programm wird über eine grafische Benutzeroberfläche gesteuert.

Die VISART-Dateien lassen sich zudem noch mit Werkzeugen durchblättern (VISARTUL) oder bearbeiten (VISARTOP), wie es im mittleren Teil des Bildes dargestellt ist.

Bild 1 veranschaulicht, wie über die Schnittstelle im Prinzip von jedem Code auf jedes Visualisierungs- oder sonstige Auswerteprogramm zugegriffen werden kann. Der Anschluß eines neuen Codes an die Schnittstelle erfordert nur die Umstellung von dessen Postprocessor-Ausgabe auf das VISART-Format, um das ganze Spektrum an Werkzeugen und Visualisierungsprogrammen nutzen zu können. Der Anschluß eines neuen Visualisierungsprogramms erfordert die einmalige Erweiterung des Adapter-Programms auf das erforderliche Eingabeformat, um es für sämtliche angeschlossene Codes zugänglich zu machen.

# 2. Der VISART-Anwendungsbereich

In diesem Kapitel werden die geometrischen, physikalischen und numerischen Annahmen dargelegt, die den Codes und Auswerteprogrammen zugrunde liegen, für die der VISART-Standard entworfen wurde. (Dieses Kapitel ist eine etwas vereinfachte Version des Kapitels 2 aus Teil 2 [1b] der VISART-Dokumentation.)

## 2.1 Räumliche Diskretisierung

### 2.1.1 Netze und Punktmengen

Anwendungs-Codes in unserem Kontext berechnen physikalische Größen über einem festen oder zeitlich variablen, üblicherweise zwei- oder dreidimensionalen räumlichen Gebiet.

Der Numerik der Codes, für die der Standard in erster Linie gedacht ist, liegt eine Überdeckung des räumlichen Gebiets mit einem **Netz** aus diskreten **Maschen** zugrunde (siehe 2.1.2). Die Codes berechnen die physikalischen Größen für jede Masche, wobei die Größen entweder der Masche als ganzer (sozusagen lokalisiert im Mittelpunkt der Masche), einem Punkt auf der Hülle der Masche oder einem **Gitterpunkt** zugeordnet sind (siehe Bild 3a und 3b).

In Maschennetzen können die einzelnen Maschen durch ihre **Maschenindizes** identifiziert werden; die Zuordnung der Größen zu einem Ort in Bezug auf die Maschen ist durch Lokationsindikatoren (siehe Bild 3a und 3b) möglich. Es genügt, die Koordinaten des Netzes zu speichern, um alle üblichen Lokationen abzudecken.

Bei Codes, die nicht mit Netzen arbeiten, sondern mit **Punktmengen** über dem räumlichen Gebiet, werden die Koordinaten der **Punkte**, an denen die physikalischen Größen berechnet werden, direkt gespeichert. (Punkte können aber auch beim Vorliegen von Maschennetzen definiert werden.)

Sowohl bei Netzen als auch bei Punktmengen kann die räumliche Dimension des Netzes bzw. der Punktmenge kleiner als die Dimension des Raums sein (1D-Netz in 2D oder 3D; 2D-Netz in 3D, z.B. eine Zylinderoberfläche).

**Definition von Begriffen**

| | | | |
|---|---|---|---|
| „Masche“ | (2D oder 3D) | ≡ 2D- bzw. 3D-Masche oder -Zelle | |
| „Kante“ | (2D oder 3D) | ≡ 2D-Rand bzw. 3D-Kante | für |
| „Hülle“ | (2D oder 3D) | ≡ 2D-Rand bzw. 3D-Hüllfläche | Maschen und |
| „Ecke“, „Gitterpunkt“ | (2D oder 3D) | ≡ Schnittpunkt von „Kanten“ | Netze |
| „Gitter“ | (2D oder 3D) | ≡ Menge aller Maschenkanten | |

### 2.1.2 Reguläre und irreguläre Netze

Die Geometrie eines Netzes kommt in zwei Hauptvarianten vor (siehe die Zeilen von Bild 2):

- Das Netz fällt mit einem der gebräuchlichen Koordinatensysteme zusammen (sog. **reguläres Netz**). Darunter fallen z.B. kartesische Koordinaten, krummlinig-orthogonale Koordinaten oder auch affine Koordinaten. Hier definieren die Koordinatenlinien/flächen die 2D-/3D-Maschen bzw. die Gitterpunkte. Die Maschen bzw. die Gitterlinien/flächen sind vierseitig (2D) bzw. sechsflächig (3D) orthogonal oder affin und ändern sich zeitlich nicht. Koordinatenlinien/flächen sind einfache Funktionen *einer* Koordinate, weshalb es genügt, die Koordinate einmal pro Richtung zu speichern. Zweckmäßigerweise werden die Koordinaten der Maschenhüllen gespeichert, aus denen sich die Koordinaten der Maschenmitten ableiten lassen.

  Diese Netze eignen sich besonders für Eulersche Diskretisierung.

- Das Netz ist nicht regulär, läßt sich aber auf ein reguläres Netz abbilden (sog. **irreguläres Netz**). Da die Netzhüllen beliebige Linien/Flächen sein können, muß jeder Maschenecke-punkt/mittelpunkt bzw. Gitterpunkt durch sein Koordinatenpaar/tripel (bezogen auf das zugrunde liegende Koordinatensystem) definiert werden. Die 2D/3D-Maschen bzw. die Gitterlinien/flächen sind nun nicht mehr notwendigerweise orthogonal oder affin, aber immer noch vierseitig (2D) bzw. sechsflächig (3D). Sie können sich im allgemeinen Fall auch zeitlich ändern. Die Koordinaten müssen hier für jede benötigte Lokation auf den Maschen bzw. für jeden Gitterpunkt gespeichert werden.

  Diese Netze eignen sich besonders für Eulersche Diskretisierung mit randangepaßten Koordinaten, usw., aber auch für Lagrangesche Diskretisierung, Rezoning, usw..

Auch andere Netzvarianten (Netze aus flächen/raumfüllenden Elementarzellen) sind denkbar; siehe hierzu Teil 2 [1b] der VISART-Dokumentation.

### 2.1.3 Volle und defektive Netze

Die äußeren Grenzen der regulären und irregulären Netze fallen in der Regel mit Koordinatenlinien/flächen zusammen bzw. können darauf abgebildet werden; die Netze sind als ganzes vierseitig (2D) bzw. sechsflächig (3D) und durch Maschen voll ausgefüllt. Wir sprechen dann von einem **vollen Netz**. Die Anzahl der Maschen ergibt sich dort aus dem Produkt der Anzahl der Maschen in jeder Koordinatenrichtung.

Im allgemeinen Fall braucht ein Netz aber nicht voll belegt sein; es kann „Löcher" enthalten, am Rande „ausgefranst" sein, aus Teilnetzen zusammengesetzt sein, oder überhaupt nur aus einigen explizit definierten Maschen bestehen (siehe die Spalten von Bild 2).

Diese Situation kann im Standard auf verschiedene Weise behandelt werden (zu „Größen" siehe 2.3):

- In einer Hilfsgröße, die über allen Maschen des vollen Netzes definiert ist, wird angegeben, ob eine Masche belegt oder leer ist, d.h. ein „Loch" enthält. Die eigentlichen Größen erhalten an den leeren Maschen bedeutungslose Werte. Aus Gründen der Speicherplatzökonomie eignet sich diese Methode nur für „schwach defektive Netze", wenn relativ wenige Maschen leer sind.
- Die belegten Teile des Netzes werden aus vierseitigen (2D) bzw. sechsflächigen (3D) Teilnetzen[1] zusammengesetzt, und die Werte der Größen nur über diesen Teilnetzen gespeichert. Diese Methode eignet sich vor allem dann, wenn nur wenige Teilnetze zur Darstellung des Netzes ausreichen.
- Im allgemeinsten Fall werden die belegten Maschen (in beliebiger Reihenfolge) durch ihre Indexpaare/tripel in Bezug auf das einschließende volle Netz explizit definiert, und die zugehörigen Werte der Größen in der gleichen Reihenfolge über diesen Maschen gespeichert. Wir bezeichnen ein so definiertes Netz als ein **defektives Netz** schlechthin. Wegen der aufwendigeren Speicherung und Handhabung eignet sich diese Methode vor allem für „stark defektive Netze", bei denen relativ viele und unregelmäßig verteilte Maschen (in Bezug auf das volle Netz) leer sind.

### 2.1.4 Teilnetze und Subnetze

Ein **Teilnetz** ist ein Ausschnitt aus einem vollen Netz, der durch Festlegung von Anfangs- und Endmaschenindizes innerhalb des Netzes in einer oder mehreren Koordinatenrichtungen entsteht (siehe Bild 2).

Fallen Anfangs- und Endmaschenindex eines Teilnetzes in einer oder mehreren Koordinatenrichtungen zusammen, so sprechen wir von einem degenerierten Teilnetz. Für node-based Darstellung läßt sich ein degeneriertes Teilnetz als ein **Subnetz** auffassen.[2] Ein Subnetz entsteht durch einen Schnitt (oder Schnitte) durch das volle Netz, d.h. durch Konstanthalten einer oder mehrerer Koordinaten des Systems. Ein Subnetz hat deshalb eine „Subdimension" kleiner als die Netzdimension (im Extremfall die Subdimension Null).

[1] Die Teilnetze (siehe 2.1.4) können ihrerseits wieder „Löcher" enthalten.

[2] In cell-based Darstellung (siehe 1.1.3) kann ein degeneriertes Teilnetz auch mit der vollen Netzdimension dargestellt werden.

### 2.1.5 Reihenfolge und Lokationen

Beim Vorliegen eines Netzes aus Maschen oder Gitterpunkten werden die Werte der Größen den Maschen bzw. Gitterpunkten zugeordnet, entweder bei vollen Netzen in einer „natürlichen“ Reihenfolge, oder bei defektiven Netzen in einer durch einen Indexvektor gegebenen Folge.

Relativ zu einer Masche können Werte an verschiedenen Stellen lokalisiert sein. Im Standard werden diese Stellen durch den Lokationsindikator spezifiziert (siehe Bild 3a und 3b). Die Lokation hat auch Einfluß auf die Anzahl der Werte und die Zählung der Maschenindizes:

- Bei Lokationen in der Maschenmitte stimmt die Anzahl der Werte mit der Anzahl der Maschen überein.
- Bei Lokationen auf Gitterpunkten ist die Anzahl der Werte in jeder Koordinatenrichtung um 1 größer als die Anzahl der Maschen.
- Bei Lokationen auf der Maschenhülle ist die Anzahl der Werte in der betreffenden Koordinatenrichtung um 1 größer als die Anzahl der Maschen, wenn auch die Netzhülle einbezogen wird.

Bei Lokationen in den Maschenmitten läuft die Zählung der Maschenindizes ab dem Index 1. Bei Lokationen auf Gitterpunkten und Maschenhüllen läuft die Zählung der Maschenindizes in der betreffenden Richtung ab dem Index 0.

Stellen auf der Netzhülle allein können im Standard ebenfalls mit entsprechenden (negativen) Lokationsindikatoren spezifiziert werden (siehe Bild 3a und 3b).[1] Bei der Übergabe der Daten an die Visualisierungssysteme können die Werte auf der Netzhülle den Werten im Netzinnern zwecks geometrisch getreuer und physikalisch vollständiger Darstellung (siehe 1.1.3) überlagert werden.

### 2.1.6 Maschenunterteilungen

Bei der Anwendung von Multigrid-Verfahren in den Codes werden die Werte physikalischer Größen in sukzessiv verfeinerter räumlicher Diskretisierung berechnet. In der Regel werden Maschen des ursprünglichen Netzes in den Maschenmitten nach den Koordinatenrichtungen unterteilt, danach die entstandenen Untermaschen auf die gleiche Weise, usw., sodaß Untermaschen mit der Hälfte, einem Viertel, einem Achtel usw. der ursprünglichen Maschenweiten entstehen. In jeder Unterteilungsstufe können den Untermaschen Werte der Größen zugeordnet werden.

Auch Werte von Größen über solchen Maschenunterteilungen lassen sich mit dem Standard spezifizieren (siehe hierzu Teil 2 [1b] der VISART-Dokumentation).

[1] Bei defektiven Netzen ist die vollständige und eindeutige Spezifizierung der Netzhülle nur über eine Hilfsgröße, den Oberflächennormalenvektor, möglich, der für die Randmaschen des Netzes definiert ist und dort für jede Lokation auf der Netzhülle (Lokation -33) Werte erhält. Dies ist in Bild 3b durch die Fahnen angedeutet, die von den Lokationen zur jeweiligen Maschenmitte weisen.

### 2.2 Zeitliche Diskretisierung

Anwendungs-Codes in unserem Kontext berechnen physikalische Größen außer über einem räumlichen Gebiet meist auch über der Zeit (hier „Problemzeit“ genannt, im Unterschied zur Rechenzeit des Computers usw.).

Der Numerik der Codes, für die der Standard in erster Linie gedacht ist, liegt meist ein Fortschreiten der Problemzeit in diskreten **Problemzeitschritten** zugrunde. Die Codes berechnen die physikalischen Größen jeweils über einen **Problemzyklus** hinweg für den nächsten **Problemzeitpunkt**, der sich aus dem jeweiligen Problemzeitschritt ergibt.

Auf eine Postprocessor-Datei werden die Größen aus Platz-, Zeit- und Kostengründen meist nicht zu *allen* sich aus den Problemzeitschritten ergebenden Problemzeitpunkten ausgegeben, sondern nur zu im Code spezifizierten Postprocessor-Problemzeitpunkten oder -Problemzyklen[1]. In einer VISART-Datei entspricht jedem solchen Postprocessor-Problemzeitpunkt oder -Problemzyklus ein Rumpf-Paket. In einem Rumpf-Paket stehen die Werte der Größen (über dem Netz, an Maschen oder Punkten oder integral) für den jeweiligen Postprocessor-Problemzeitpunkt bzw. -Problemzyklus. Alternativ oder zusätzlich können aber auch die Werte von Größen zu allen Problemzeitpunkten (oder einer Teilmenge davon), die seit dem letzten Postprocessor-Problemzeitpunkt durchlaufen wurden, in ihrer zeitlichen Folge aneinandergereiht, in einem Rumpf-Paket stehen[2].

[1] Als erster Postprocessor-Problemzeitpunkt wird zweckmäßigerweise Null genommen, entsprechend dem Problemzyklus Null.

[2] Dies wird man zweckmäßigerweise nur für ausgewählte integrale Größen oder für Größen an ausgewählten Punkten oder Maschen tun.

## 2.3 Größen

### 2.3.1 Konstante und variable Größen

In der VISART-Datei können zeitlich konstante und zeitlich variable physikalische Größen oder andere Parameter des ausgeführten Codes stehen.

Zeitlich konstante Größen sind z.B. geometrische Größen zeitlich konstanter Netze (Maschenweiten, Maschenareale, Maschenvolumina, Abschrägungen von Randmaschen, Maschenbelegungen bei Löchern in Netzen, Oberflächennormalen) oder zeitlich konstante Punktkoordinaten, ferner zeitlich konstante Modellstrukturen und andere Modellparameter, Anfangswerte und zeitlich konstante Randwerte des Problems, usw..

Zeitlich variable Größen sind die obigen Größen, falls sie nicht zeitlich konstant sind, sowie die über der Problemzeit berechneten physikalischen Größen wie Massen, Dichten, Volumenanteile, Drücke, Geschwindigkeiten, Temperaturen, Energien, oder die Orte von Phasengrenzen, Medienoberflächen, Regimearten; entweder als Feldgrößen oder über das Gebiet integriert.

Konstante Größen stehen zweckmäßigerweise im Kopf-Paket der Datei, variable Größen stehen in den Rumpf-Paketen der Datei.

### 2.3.2 Datentyp

Die Daten der physikalischen Größen sind vom Typ `REAL` oder `INTEGER`. Daneben kommen Daten vom Typ `CHARACTER` vor (z.B. für Identifikationen in Gruppen, die als Inhaltsverzeichnis dienen), sowie vom Typ `LOGICAL` (z.B. bei relationalen und logischen Verknüpfungen von Größen im Bearbeitungsprogramm VISARTOP (siehe Teil 4 [1d] der VISART-Dokumentation)).

### 2.3.3 Skalare und Vektoren

Geometrische und physikalische Größen können skalar oder vektoriell sein. Vektorielle Größen sind z.B. Koordinatenpaare/tripel von Punkten oder von Lokationen irregulärer Netze, Maschenindexpaare/tripel zur Spezifikation defektiver Netze, Abschrägungs- und Oberflächennormalen, Maschenareale, Maschenweiten, Geschwindigkeiten und andere vektorielle Feldgrößen.

Vektorkomponenten beziehen sich immer auf die entsprechenden Koordinaten des jeweiligen Koordinatensystems, d.h. die Komponenten liegen in Richtung der Koordinatenlinien. (Man beachte, daß bei irregulären Koordinaten die Koordinatenlinien im allgemeinen nicht mit den Gitterlinien des Netzes zusammenfallen!) Die Beträge der Vektorkomponenten werden aber nicht unbedingt in der Einheit der jeweiligen Koordinate angegeben, sondern immer in der Einheit der jeweiligen Größe.

Die Vektordimension muß nicht mit der Dimension des Netzes bzw. der Punktmenge übereinstimmen, sondern sie kann größer sein, wenn die Dimension des Einbettungsraums größer ist.

### 2.3.4 Netz-, Maschen- bzw. Punkt- und Integralgrößen

Ortsabhängige Größen sind als **Netzgrößen** oder **Maschengrößen** für alle Maschen des jeweiligen Netzes beschrieben. Darunter fallen die meisten nicht-diskreten physikalischen Größen. **Punktgrößen**, wie z.B. die Eigenschaften von Teilchen, sind dagegen nur an den explizit definierten Punkten im Netz definiert, die sich zudem meist noch mit der Problemzeit ändern.

Ortsunabhängige oder über das gesamte Gebiet integrierte ortsabhängige Größen heißen **Integralgrößen**. Dies sind z.B. Gesamtvolumen, -masse, -energie des Feldes, aber auch Modelloptionen, Eingabeparameter, usw..

# 3. Der VISART-Standard

Dieses Kapitel führt in den Aufbau, das Format und den Inhalt einer VISART-Datei ein. Ausführliche und vollständige Information dazu findet man im Teil 2 [1b] der VISART-Dokumentation.

## 3.1 Die VISART-Datei

### 3.1.1 Der VISART-Dateiaufbau

Die VISART-Datei ist eine **sequentielle** Datei. Ihre Daten werden normalerweise unformatiert (d.h. in maschineninterner, binärer Darstellung) gespeichert, können jedoch auch formatiert (d.h. in ASCII Character-Darstellung) abgespeichert werden (z.B. weil die Datei zwischen Maschinen mit unterschiedlicher maschineninterner Zeichendarstellung ausgetauscht werden soll).

Die Information auf der Datei ist formal gegliedert in **Pakete**, diese in **Gruppen** von Sätzen, und diese wieder in (logische) **Sätze**.

Den Anfang der Datei bildet das **Kopf**-Paket mit den konstanten (problemzeitunabhängigen) Daten des gerechneten Problems. Dann folgen beliebig viele **Rumpf**-Pakete, jeweils mit Daten für einen bestimmten Problemzyklus oder -zeitpunkt.

Sowohl Kopf-Paket als auch Rumpf-Pakete bestehen aus **obligatorischen**, jeweils einmal vorkommenden, Gruppen und aus **fakultativen**, auch mehrfach vorkommenden, Gruppen. Das Vorkommen der letzteren wird nur vom Code und vom gerechneten Problem bestimmt. Abgesehen von den obligatorischen Beschreibungs- und Geometriegruppen erlauben die Formate der vorgesehenen Gruppen sowohl im Kopf-Paket als auch in den Rumpf-Paketen das Speichern von Größen (skalaren oder vektoriellen), die allen Maschen des Netzes zugeordnet sind, die ausgewählten Maschen oder Punkten des Gebiets zugeordnet sind, oder die raumunabhängig („integral“) sind. In den Rumpf-Paketen ist außerdem noch ein Format für über alle Problemzeiten seit dem Zeitpunkt des letzten Rumpf-Pakets angesammelten Folgen der drei obigen Größenarten vorgesehen.

Der erste Satz jeder Gruppe dient als **Kennsatz**. Er beginnt mit einer Integer-Gruppenkennzahl; als nächstes folgt eine Integer-Zahl, die die Anzahl der nachfolgenden Records (siehe 3.1.2) der Gruppe angibt; danach folgt eine Zeichenkette mit der code- und problemeigenen Identifikation der betreffenden Größe. Drei weitere Integer-Zahlen geben gewöhnlich Länge der Datensätze, Skalar- oder Vektoreigenschaft und Datentyp der Größe an. Außer bei den Kopf-Beschreibungsgruppen (Gruppenkennzahl 0, ..., 3) ist der Kennsatz jeder Gruppe von konstanter Länge. Damit ist ein unkompliziertes Durchsuchen der Datei möglich. Als zweiter Satz steht in manchen Fällen ein **Spezifikationssatz** mit weiteren Angaben. Danach kommen ein oder mehrere **Datensätze**.

Weitere Details findet man im Teil 2 [1b] der VISART-Dokumentation. Die nachfolgende Liste 1 zeigt die Struktur einer VISART-Datei.

**Liste 1: Struktur einer VISART-Datei**

**Kopf-Paket:** problemzeitunabhängige Daten
- Gruppe 0: Datei-Information
- Gruppe 1: Code-Information
- Gruppe 2: Prozess-Information
- Gruppe 3: Problem-Information
- Gruppe 4: Geometrie des Netzes
- fakultative Gruppen 5 (Netzkonstanten)
- fakultative Gruppen 6, 7 (Maschen- oder Punktkonstanten)
- fakultative Gruppen 9 (Integralkonstanten)

**Rumpf-Paket:** problemzeitabhängige Daten zur Zeit $t_0$
- Gruppe 10: Zyklus des Pakets (Problemzeit und -zyklus)
- fakultative Gruppen 15 (Netzvariable)
- fakultative Gruppen 16, 17 (Maschen- oder Punktvariable)
- fakultative Gruppen 19 (Integralvariable)
- fakultative Gruppen 20, 21 (Zeitfunktionsvariable)

**Rumpf-Paket:** problemzeitabhängige Daten zur Zeit $t_1$
- Gruppe 10: Zyklus des Pakets (Problemzeit und -zyklus)
- fakultative Gruppen 15 (Netzvariable)
- fakultative Gruppen 16, 17 (Maschen- oder Punktvariable)
- fakultative Gruppen 19 (Integralvariable)
- fakultative Gruppen 20, 21 (Zeitfunktionsvariable)

**Rumpf-Paket:** problemzeitabhängige Daten zur Zeit $t_2$
- Gruppe 10: Zyklus des Pakets (Problemzeit und -zyklus)
- fakultative Gruppen 15 (Netzvariable)
- fakultative Gruppen 16, 17 (Maschen- oder Punktvariable)
- fakultative Gruppen 19 (Integralvariable)
- fakultative Gruppen 20, 21 (Zeitfunktionsvariable)

. . . . . .

Das erste Rumpf-Paket enthält normalerweise die Anfangswerte der Größen zu Beginn einer Rechnung (die Zeitfunktionsvariablenfolgen bestehen dort nur aus *einem* Glied). Das letzte Rumpf-Paket enthält normalerweise die Werte der Größen am Ende einer Rechnung (sei es wegen Erreichens einer vorgegebenen Problemzeit oder wegen Abbruch der Rechnung aus verschiedenen Gründen). In der Regel werden die Rumpf-Pakete (außer dem letzten) in gleichmäßigen Zeit- oder Zyklusabständen aufeinanderfolgen.

Die Rumpf-Pakete brauchen nicht alle gleich aufgebaut sein, werden dies aber in der Regel sein (so verlangen es auch die VISART-Werkzeuge und -Adapter-Programme; siehe 4.).

Denkbar ist auch eine Überlagerung mehrerer Folgen von Rumpf-Paketen, mit jeweils eigenen Zeitabständen und Zusammensetzungen, die sich durch ihre Zyklus-Identifikationen unterscheiden müssen.

### 3.1.2 Die VISART-Gruppenformate

Die Gruppen werden formal durch ihre Gruppenkennzahl identifiziert. Die Gruppen 0, 1, 2, 3 (Liste 2) dienen zur Beschreibung der Datei, des Codes, des Prozesses und des gerechneten Problems. Die Gruppe 4 (Liste 3) definiert die Geometrie und evtl. die Koordinaten des Problems. Die Gruppe 10 (Liste 4) definiert Problemzeit und Problemzyklus eines Rumpf-Pakets. Alle anderen Gruppen beschreiben Größen, entweder zeitunabhängige im Kopf-Paket (Gruppen 5, 6, 7, 9), oder zeitabhängige in den Rumpf-Paketen (Gruppen 15, 16, 17, 19, 20, 21), zu den Problemzeiten des jeweiligen Pakets. Die Gruppen 5/15 (Liste 5) sind für Größen über dem vollen Netz (auch mit Löchern) oder über Teilnetzen desselben gedacht, die Gruppen 6/16 und 7/17 (Liste 6) für Größen über defektiven Netzen oder in einzelnen Punkten, die Gruppen 9/19 (Liste 7) für ortsunabhängige Größen, und die Gruppen 20 und 21 (Liste 8) für Größen an bestimmten Maschen oder Punkten oder ortsunabhängige Größen, in über die Problemzeiten seit dem Zeitpunkt des letzten Rumpf-Pakets angesammelten Folgen. Die Gruppen 7, 17 und 21 dürfen nur auf eine vorangegangene Gruppe 6, 16 bzw. 20 folgen und heißen deshalb auch **Subgruppen**. Die Gruppen 6, 16 bzw. 20 heißen auch die **Referenzgruppen** der zugeordneten Subgruppen.

(Für Größen über Netzen mit mehrfach unterteilten Maschen gibt es ferner die Gruppen 51, 61, 71, 151, 161 und 171; siehe dazu Teil 2 [1b] der VISART-Dokumentation.)

In den nachfolgenden Listen 2 bis 8 sind für die jeweiligen Gruppen die Sätze, aus denen sie aufgebaut sind, und deren Inhalte aufgeführt. Bei der Beschreibung des Inhalts entspricht jede Zeichenkette einer `INTEGER`-, `REAL`- oder `CHARACTER`-Variablen. Mit `I` beginnende Ketten bezeichnen `INTEGER`-Variable; mit `C` beginnende Ketten bezeichnen `CHARACTER`-Variable. Alle anderen Ketten bezeichnen Variable, die `REAL` und—je nach Fall—auch `INTEGER`, `LOGICAL` oder `CHARACTER` -Variable sein können.

Die mit `m` bezeichnete Variable in den Kennsätzen der Gruppen steht immer für die Anzahl der nachfolgenden **Records**[1] der Gruppe oder Subgruppe. Bei unformatierten Dateien ist `m` gleich der Anzahl der (logischen) Sätze. Bei formatierten Dateien ergibt sich `m` aus der Aufteilung des (logischen) Satzes in Records von maximal 80 Zeichen. Dabei werden `INTEGER`-Variable im Fortran-Format `I8`, `LOGICAL`-Variable im Format `L8`, `REAL`-Variable im Format `E16.8` und `CHARACTER`-Variable im Format `A8` dargestellt. (Alle Kennsätze und Spezifikationssätze passen dann in *einen* Record.)

Die Werte der Indikatoren in den Kennsätzen und Spezifikationssätzen (für Dimensionen, Netz, Koordinatensystem, Lokation, Reihenfolge, Datentyp und Teilnetzeinschränkung) müssen den Tabellen im Teil 2 [1b] der VISART-Dokumentation entnommen werden.

[1] mit „Records“ bezeichnen wir hier physische Sätze, so wie sie in Fortran mit *einem* `READ/WRITE`-Befehl gelesen, geschrieben oder übersprungen werden können, ohne daß man die Länge der Sätze kennen muß. (Bei Implementierungen in anderen Sprachen gibt es evtl. keinen entsprechenden Begriff.)

**Liste 2: Struktur der Beschreibungsgruppen (Gruppen 0 bis 3)**

Diese Gruppen (am Anfang der Datei, im Kopf-Paket) beschreiben die Art der Datei, sowie den Code und dessen Binärmodul, den Prozess und die Code-Eingabe für das Problem, die die Daten produzierten.

### File-Gruppe (Gruppe 0)

**Kennsatz:** `0,IDDBL,CDRELS`

`IDDBL`: Indikator für einfache oder doppelte Genauigkeit der Gleitkommazahlen
`CDRELS`: Release des VISART-Standards, der der Datei zugrunde liegt.

### Code-Gruppe (Gruppe 1)

**Kennsatz:** `1,0,CCNAME,CCRELS,CCAUTH,CCDATE,CCTIME`

`CCNAME`: Name des ausgeführten Codes
`CCRELS`: Release des ausgeführten Codes
`CCAUTH`: Ersteller des ausgeführten Binärprogramms
`CCDATE`: Erstellungsdatum des ausgeführten Binärprogramms
`CCTIME`: Erstellungsuhrzeit des ausgeführten Binärprogramms

### Prozess-Gruppe (Gruppe 2)

**Kennsatz:** `2,0,CJNAME,CJNUMB,CJAUTH,CJDATE,CJTIME`

`CJNAME`: Name der Workstation für die Programmausführung
`CJNUMB`: Prozessnummer der Programmausführung
`CJAUTH`: Absender der Programmausführung
`CJDATE`: Startdatum der Programmausführung
`CJTIME`: Startuhrzeit der Programmausführung

### Problem-Gruppe (Gruppe 3)

**Kennsatz:** `3,2,CINAME,CINUMB,CIAUTH,CIDATE,CITIME`

`CINAME`: im Erstlauf: leer; im Restart-Lauf: Workstation-Name vom Erstlauf
`CINUMB`: im Erstlauf: Name der Eingabedatei; im Restart-Lauf: Prozessnummer vom Erstlauf
`CIAUTH`: im Erstlauf: Ersteller der Eingabedatei; im Restart-Lauf: Absender vom Erstlauf
`CIDATE`: im Erstlauf: Versionsdatum der Ed.; im Restart-Lauf: Startdatum vom Erstlauf
`CITIME`: im Erstlauf: Versionsuhrzeit der Ed.; im Restart-Lauf: Startuhrzeit vom Erstlauf

**Problemname:** `CIIDE1(1),...,CIIDE1(10)`

`CIIDE1`: Teil 1 des Namens des eingegebenen Problems

**Problemname:** `CIIDE2(1),...,CIIDE2(10)`

`CIIDE2`: Teil 2 des Namens des eingegebenen Problems

**Liste 3: Struktur der Geometrie-Gruppe (Gruppe 4)**

Die Geometrie-Gruppe (im Kopf-Paket) definiert die Geometrie des Netzes und des einbettenden Raums, das verwendete Koordinatensystem, die Anzahl der Maschen in den Koordinatenrichtungen, und ggf. die Koordinatenwerte der Maschenhüllen oder Maschenmitten.

*Für reguläre Netze:*

**Kennsatz:** `4,m,CZNAME,IZDIM,IZGEO,IZSYS`

`m`: Anzahl der nachfolgenden Sätze
`CZNAME`: Identifikation der Gruppe (hier `GEOMETRY`)
`IZDIM`: Indikator für die Dimension `IZMSH` des Netzes und die Dimension `IZSPC` des Raumes
`IZGEO`: Indikator für ein reguläres Netz (`1`)
`IZSYS`: Indikator für das Koordinatensystem

**Spezifikationssatz:** `IZNOI,IZNOJ,IZNOK,IZLOC,ZANGI,ZANGJ,ZANGK`

`IZNOI`: Anzahl der Koordinatenwerte in $i$-Richtung des Netzes
`IZNOJ`: Anzahl der Koordinatenwerte in $j$-Richtung des Netzes
`IZNOK`: Anzahl der Koordinatenwerte in $k$-Richtung des Netzes
`IZLOC`: Lokationsindikator (Maschenmitten, Maschenhüllen oder Gitterpunkte)

**Datensatz:** `ZKORI(1),...,ZKORI(IZNOI)`

`ZKORI`: Maschenkoordinaten in $i$-Richtung des Netzes

Wenn `IZMSH` $> 1$:
**Datensatz:** `ZKORJ(1),...,ZKORJ(IZNOJ)`

`ZKORJ`: Maschenkoordinaten in $j$-Richtung des Netzes

Wenn `IZMSH` $> 2$:
**Datensatz:** `ZKORK(1),...,ZKORK(IZNOK)`

`ZKORK`: Maschenkoordinaten in $k$-Richtung des Netzes

*Für irreguläre Netze:*

**Kennsatz:** `4,m,CZNAME,IZDIM,IZGEO,IZSYS`

`m`: Anzahl der nachfolgenden Sätze (hier `1`)
`CZNAME`: Identifikation der Gruppe (hier `GEOMETRY`)
`IZDIM`: Indikator für die Dimension `IZMSH` des Netzes und die Dimension `IZSPC` des Raumes
`IZGEO`: Indikator für ein irreguläres Netz (`3`)
`IZSYS`: Indikator für das Koordinatensystem

**Spezifikationssatz:** `IZNOI,IZNOJ,IZNOK,IZLOC,ZANGI,ZANGJ,ZANGK`

`IZNOI`: Anzahl der Koordinatenwerte in $i$-Richtung des Netzes
`IZNOJ`: Anzahl der Koordinatenwerte in $j$-Richtung des Netzes
`IZNOK`: Anzahl der Koordinatenwerte in $k$-Richtung des Netzes
`IZLOC`: Lokationsindikator (Maschenmitten, Maschenhüllen oder Gitterpunkte)

(Die Maschenkoordinaten werden in diesem Fall in Gruppen 5/15, zweckmäßigerweise mit der Identifikation `...COORD...`, für die jeweils benötigten Maschenlokationen angegeben.)

**Liste 4: Struktur der Zyklus-Gruppe (Gruppe 10)**

Die Zyklus-Gruppe (am Anfang jedes Rumpf-Pakets) definiert den Problemzeitpunkt und -zyklus des Rumpf-Pakets.

*Wenn unformatierte Datei:*

**Zyklus-Gruppe:** `10,0,CYNAME,IYCC,YTIME,IDUM`
- `CYNAME`: Identifikation der Gruppe (hier `CYCL....`)
- `IYCC`: Problemzyklusnummer
- `YTIME`: Problemzeitpunkt
- `IDUM`: 0 (ohne Bedeutung)

*Wenn formatierte Datei:*

**Zyklus-Gruppe:** `10,0,CYNAME,IYCC,YTIME`
- `CYNAME`: Identifikation der Gruppe (hier `CYCL....`)
- `IYCC`: Problemzyklusnummer
- `YTIME`: Problemzeitpunkt

**Liste 5: Struktur der Netzkonstanten/variablen-Gruppen (Gruppen 5/15)**

Die Netzkonstanten/variablen-Gruppen beschreiben Größen über dem vollen Netz (auch mit Löchern) oder über Teilnetzen desselben. Die Netzkonstanten-Gruppe 5 im Kopf-Paket beschreibt zeitlich konstante Größen, die Netzvariablen-Gruppe 15 in den Rumpf-Paketen beschreibt zeitlich variable Größen zu den jeweiligen Problemzeiten des Pakets.

Die Strukturen der Gruppen 5 und 15 (letztere wird im folgenden gezeigt) sind bis auf die Gruppenkennzahl identisch.

**Kennsatz:** `15,m,CSNAME,ISNO,ISKOM,ISTYP`

`m`: Anzahl der nachfolgenden Sätze
`CSNAME`: Identifikation der Größe
`ISNO`: Anzahl der Werte der Größe
`ISKOM`: Vektordimension der Größe
`ISTYP`: Indikator für Integer-, Real-, Logical- oder Character-Daten

**Spezifikationssatz:**
`ISDIM,ISPRT,ISKORI1,ISKORI2,ISKORJ1,ISKORJ2,ISKORK1,ISKORK2,ISORD,ISLOC`

`ISDIM`: 0
`ISPRT`: Teilnetzindikator (0 wenn kein Teilnetz)
`ISKORI1, ISKORI2`: Randmaschen in $i$-Richtung für Teilnetz (sonst 0)
`ISKORJ1, ISKORJ2`: Randmaschen in $j$-Richtung für Teilnetz (sonst 0)
`ISKORK1, ISKORK2`: Randmaschen in $k$-Richtung für Teilnetz (sonst 0)
`ISORD`: Reihenfolgeindikator
`ISLOC`: Lokationsindikator

**Datensatz:** `SKOMI(1),...,SKOMI(ISNO)`

`SKOMI`: Werte (skalar oder $i$-Komponente)

Wenn `ISKOM` $> 1$:
**Datensatz:** `SKOMJ(1),...,SKOMJ(ISNO)`

`SKOMJ`: Werte ($j$-Komponente)

Wenn `ISKOM` $> 2$:
**Datensatz:** `SKOMK(1),...,SKOMK(ISNO)`

`SKOMK`: Werte ($k$-Komponente)

Beispiele für Größen, die mit Gruppen 5 und 15 beschrieben werden:

- Gruppen 5 (im Kopf-Paket): zeitunabhängige Koordinatenvektoren irregulärer Netze; Maschenbelegung bei Netzen mit Löchern; zeitunabhängige Randwerte (auf der Netzhülle) zeitabhängiger Größen.
- Gruppen 15 (in den Rumpf-Paketen): zeitabhängige skalare oder vektorielle Größen in den Maschenmitten (z.B. Drücke, Dichten, Volumenanteile, Geschwindigkeitsvektoren) oder in den Maschenmitten und zusätzlich auf der Netzhülle; oder Größen auf den Maschenhüllen (z.B. Geschwindigkeitskomponenten bei versetzten Maschen).

**Liste 6: Struktur der Maschen- (oder Punkt-)konstanten/variablen-Gruppen (Gruppen 6/16 und Subgruppen 7/17)**

Die Maschen- (oder Punkt-)konstanten/variablen-Gruppen beschreiben Größen über defektiven Netzen oder in einzelnen Punkten eines Netzes. Die Gruppen 6/16 spezifizieren die Maschen oder Punkte, die Subgruppen 7/17 enthalten die Werte der Größen an den Maschen oder Punkten. Die Maschen- (oder Punkt-)konstanten-Gruppen 6 und 7 im Kopf-Paket beschreiben zeitlich konstante Größen, die Maschen- (oder Punkt-)variablen-Gruppen 16 und 17 in den Rumpf-Paketen beschreiben zeitlich variable Größen zu den jeweiligen Problemzeiten des Pakets.

Die Strukturen der Gruppen 6 und 16 (letztere wird im folgenden gezeigt), sowie der Subgruppen 7 und 17 (letztere wird im folgenden gezeigt), sind bis auf die Gruppenkennzahl identisch.

**Kennsatz:** `16,m,CPNAME,IPNO,IDUM,IPTYP`
`m`: Anzahl der nachfolgenden Sätze
`CPNAME`: Identifikation der Gruppe (hier z.B. `..INDEX..`)
`IPNO`: Anzahl der Werte der Gruppe und der Subgruppen
`IDUM`: 0 (ohne Bedeutung)
`IPTYP`: Indikator für Maschenindizes, Punktkoordinaten oder leere Datensätze

**Datensatz:** `PKORI(1),...,PKORI(IPNO)`
`PKORI`: Werte (skalar oder $i$-Komponente)

Wenn `IZMSH` $> 1$:
**Datensatz:** `PKORJ(1),...,PKORJ(IPNO)`
`PKORJ`: Werte ($j$-Komponente)

Wenn `IZMSH` $> 2$:
**Datensatz:** `PKORK(1),...,PKORK(IPNO)`
`PKORK`: Werte ($k$-Komponente)

**Kennsatz:** `17,m,CPNAM1,IPLOC1,IPKOM1,IPTYP1`
`m`: Anzahl der nachfolgenden Sätze
`CPNAM1`: Identifikation der Größe
`IPLOC1`: Lokationsindikator
`IPKOM1`: Vektordimension der Größe
`IPTYP1`: Indikator für Integer-, Real-, Logical- oder Character-Daten

**Datensatz:** `PKOMI1(1),...,PKOMI1(IPNO)`
`PKOMI1`: Werte (skalar oder $i$-Komponente)

Wenn `IPKOM1` $> 1$:
**Datensatz:** `PKOMJ1(1),...,PKOMJ1(IPNO)`
`PKOMJ1`: Werte ($j$-Komponente)

Wenn `IPKOM1` $> 2$:
**Datensatz:** `PKOMK1(1),...,PKOMK1(IPNO)`
`PKOMK1`: Werte ($k$-Komponente)

**Kennsatz:** `17,m,CPNAM2,IPLOC2,IPKOM2,IPTYP2`
usw.

.....

Beispiele für Größen, die mit Gruppen 6 und 16 beschrieben werden:

- Gruppen 6 (im Kopf-Paket): Maschenindexvektoren für die Maschen defektiver Netze; Maschenindexvektoren für die Netzhülle defektiver Netze.
- Gruppen 16 (in den Rumpf-Paketen): Koordinatenvektoren von Partikeln.

Beispiele für Größen, die mit Subgruppen 7 und 17 beschrieben werden:

- Subgruppen 7 (im Kopf-Paket): Oberflächennormalenvektoren für die Netzhülle defektiver Netze; zeitunabhängige Randwerte (auf der Netzhülle) zeitabhängiger Größen in defektiven Netzen.
- Subgruppen 17 (in den Rumpf-Pakete): zeitabhängige skalare oder vektorielle Größen in den Maschenmitten (z.B. Drücke, Dichten, Volumenanteile, Geschwindigkeitsvektoren) oder auf den Maschenhüllen (z.B. Geschwindigkeitskomponenten bei versetzten Maschen) in defektiven Netzen; zeitabhängige skalare oder vektorielle Größen auf der Netzhülle defektiver Netze; zeitabhängige Eigenschaften von Partikeln (z.B. Massen, Geschwindigkeiten).

**Liste 7: Struktur der Integralkonstanten/variablen-Gruppen (Gruppen 9/19)**

Die Integralkonstanten/variablen-Gruppen beschreiben integral für das ganze Netz oder sonst ortsunabhängig definierte Größen. Die Integralkonstanten-Gruppe 9 im Kopf-Paket beschreibt zeitlich konstante Größen, die Integralvariablen-Gruppe 19 in den Rumpf-Paketen beschreibt zeitlich variable Größen zu den jeweiligen Problemzeiten des Pakets.

Die Strukturen der Gruppen 9 und 19 (letztere wird im folgenden gezeigt) sind bis auf die Gruppenkennzahl identisch.

**Kennsatz:** `19,m,CGNAME,IGNO,IGKOM,IGTYP`
`m`: Anzahl der nachfolgenden Sätze
`CGNAME`: Identifikation der Gruppe
`IGNO`: Anzahl der Größen in der Gruppe
`IGKOM`: Vektordimension der Größen
`IGTYP`: Indikator für Integer-, Real-, Logical- oder Character-Daten

**Datensatz:** `GKOMI(1),...,GKOMI(IGNO)`
`GKOMI`: Werte der Größen (skalar oder $i$-Komponente)

Wenn `IGKOM` $> 1$:
**Datensatz:** `GKOMJ(1),...,GKOMJ(IGNO)`
`GKOMJ`: Werte der Größen ($j$-Komponente)

Wenn `IGKOM` $> 2$:
**Datensatz:** `GKOMK(1),...,GKOMK(IGNO)`
`GKOMK`: Werte der Größen ($k$-Komponente)

Beispiele für Größen, die mit Gruppen 9 und 19 beschrieben werden:

- Gruppen 9 (im Kopf-Paket): Identifikationen (als Character-Konstanten) von Größen, deren Werte in den zugeordneten Gruppen 19 (an gleicher Position) stehen.
- Gruppen 19 (in den Rumpf-Paketen): zeitabhängige skalare oder vektorielle Größen für das ganze Netz (z.B. Gesamtmassen, Gesamtenergien), oder netzunabhängige Größen (z.B. Programmparameter).

**Liste 8: Struktur der Zeitfunktions-Gruppen (Gruppe 20 und Subgruppe 21)**

Die Zeitfunktionsgruppen beschreiben Größen für eine Folge von Problemzeitpunkten in den Intervallen zwischen den Problemzeiten der Rumpf-Pakete. Die Gruppe 20 spezifiziert die Problemzeitpunkte oder -zyklen im Intervall bis zur jeweiligen Problemzeit des Pakets, die Subgruppen 21 enthalten die zugehörigen Werte der Größen.

**Kennsatz:** `20,m,CFNAME,IFNO,IDUM,IFTYP`

`m`: Anzahl der nachfolgenden Sätze
`CFNAME`: Identifikation der Gruppe (z.B. `TIME...`)
`IFNO`: Anzahl der Werte der Gruppe und der Subgruppen (d.h. Länge der Folgen)
`IDUM`: 0 (ohne Bedeutung)
`IFTYP`: Indikator für Problemzyklusnummern oder -zeitpunkte

**Datensatz:** `FTIME(1),...,FTIME(IFNO)`

`FTIME`: Werte

**Kennsatz:** `21,m,CFNAM1,IFKND1,IFKOM1,IFTYP1`

`m`: Anzahl der nachfolgenden Sätze
`CFNAM1`: Identifikation der Größe
`IFKND1`: Indikator für Netz/Maschengröße, Punktgröße oder Integralgröße
`IFKOM1`: Vektordimension der Größe
`IFTYP1`: Indikator für Integer-, Real-, Logical- oder Character-Daten

**Spezifikationssatz:** `FKORI1,FKORJ1,FKORK1,IFLOC1`

`FKORI1`: Maschenindex bzw. Punktkoordinate in $i$-Richtung des Netzes
`FKORJ1`: Maschenindex bzw. Punktkoordinate in $j$-Richtung des Netzes
`FKORK1`: Maschenindex bzw. Punktkoordinate in $k$-Richtung des Netzes
(jeweils Integer für Netz/Maschengröße, Real für Punktgröße, Integer-Null für Integralgröße)
`IFLOC1`: Lokationsindikator (bei Netz/Maschengröße) oder 0

**Datensatz:** `FKOMI1(1),...,FKOMI1(IFNO)`

`FKOMI1`: Werte (skalar oder $i$-Komponente)

Wenn `IFKOM1` $> 1$:
**Datensatz:** `FKOMJ1(1),...,FKOMJ1(IFNO)`

`FKOMJ1`: Werte ($j$-Komponente)

Wenn `IFKOM1` $> 2$:
**Datensatz:** `FKOMK1(1),...,FKOMK1(IFNO)`

`FKOMK1`: Werte ($k$-Komponente)

**Kennsatz:** `21,m,CFNAM2,IFKND2,IFKOM2,IFTYP2`

usw.

.....

Beispiele für Größen, die mit Subgruppe 21 beschrieben werden können:

- alle Größen der Gruppen 15 und 17 an bestimmten Maschenindizes oder Punktkoordinaten, und alle Größen der Gruppe 19.

### 3.2 Der VISART-Dateiinhalt

Der Inhalt (d.h. Art, Umfang, Bedeutung, usw.) der im VISART-Dateiformat gespeicherten Information ist durch den Standard, außer im Falle der obligatorischen Gruppen, nicht festgelegt. Er wird ausschließlich durch den Anwendungs-Code und das gerechnete Problem bestimmt. Zur Identifizierung der in den Gruppen gespeicherten Größen dient im wesentlichen die Zeichenkette der Identifikationen in den Kennsätzen der Gruppen.

Den Visualisierungsprogrammen andererseits müssen per Eingabe oder sonst die Gruppenkennzahlen und Identifikationen der aus der Datei auszuwählenden und darzustellenden Größen mitgeteilt werden.

Obwohl der Standard über die Größen und ihre Identifikationen nichts festlegt, hat es sich als zweckmäßig erwiesen, für häufig vorkommende Größen, die z.B. die Geometrie bestimmen, immer die gleichen Identifikationen zu verwenden, um die Übersicht und Verarbeitung zu erleichtern. In der nachfolgenden Tabelle werden einige Identifikationen für geometrische und andere Größen, sowie für Zyklus- und Zeitfunktionsgruppen, vorgeschlagen, wie sie z.B. auch von den VISART-Dienstprogrammen (siehe Kapitel 4) vorausgesetzt werden.

Im Anhang B des Teils 2 [1b] der VISART-Dokumentation findet man weitere Richtlinien für den Dateiaufbau und -inhalt.

**Identifikationen von obligatorischen Gruppen:**

| Gruppe | Identifikation | Beispiele |
|---|---|---|
| 4 | `GEOMETRY` | |
| 10 | `CYCL....` | `CYCLINIT` im ersten Zyklus<br>`CYCLPOST` im programmierten Ausgabezyklus<br>`CYCLFINI` bei Problemende<br>`CYCLTERM` bei Jobabbruch wegen CPU-Zeit-Ausschöpfung<br>`CYCLFAIL` bei Jobabbruch wegen Fehler |

**Identifikationen von fakultativen Gruppen:**

| Gruppe | Identifikation | Inhalt | Beispiele |
|---|---|---|---|
| 5,15 | `...COORD...` | (irreguläre) Maschenkoordinaten | `COORDN, COORDC` |
| 5,15 | `...DEFCT...` | Maschenbelegung<br>(0 für belegte Masche, 1 für Loch) | `DEFCTC` |
| 6,16 | `...INDEX...` | Maschenindizes | `INDEX, INDEXN, INDEXC` |
| 7,17 | `...NORMV...` | Normalenvektor für die Netzhülle | `S_NORMV` |
| 5 | `WIDTHS` | Maschenweiten | |
| 5 | `AREALS` | Maschenareale | |
| 5 | `VOLUMES` | Maschenvolumina | |
| 5 | `SHAPE` | irreguläre Oberflächenform von Randmaschen | |
| 9 | `CELL__NM` | Namen von Netzgrößen | |
| 9 | `......NM` | Namen von Integralgrößen | `INTGRLNM` |
| 19 | `......VL` | Werte von Integralgrößen | `INTGRLVL` |
| 20 | `TIME....` | Problemzeiten | `TIMEALL` |
| 20 | `NUMB....` | Zyklusnummern | `NUMBALL` |

# 4. Die VISART-Dienste

Dieses Kapitel stellt die grafische Benutzeroberfläche vor, mit der die Werkzeuge VISARTUL und VISARTOP und die Adapter-Programme VISAPTER und VISARVOL aufgerufen und gesteuert werden können. Im Falle von VISAPTER und VISARVOL können über die grafische Benutzeroberfläche auch die verfügbaren Visualisierungssysteme angewählt werden. Die Fähigkeiten der einzelnen Dienste können hier nur skizziert werden; vollständige Informationen dazu findet man im Teil 3 [1c] (VISAPTER), Teil 4 [1d] (VISARTOP) und Teil 5 [1e] (VISARVOL) der VISART-Dokumentation.

## 4.1 Betriebssysteme und Programmiersprachen

Wie in Kapitel 1 dargelegt, ist das VISART-Konzept unabhängig von Betriebssystemen und Programmiersprachen.

Die hier vorgestellten VISART-Werkzeuge und -Adapter-Programme sind in Fortran geschrieben und damit in jedem Betriebssystem implementierbar. Sie laufen zur Zeit unter UNIX auf IBM-RS/6000-Workstations (sowie unter MVS auf IBM-Großrechnern, wofür sie ursprünglich entworfen wurden). Alle Programme sind für den Stapelbetrieb entworfen; sie erwarten eine komplette Steuereingabe in einer Datei und laufen vom Start bis zum Programmende oder Fehlerabbruch ohne Wechselwirkung mit dem Benutzer. Die Steuereingaben der Programme sind in deren Dokumentation (s.o.) im Detail beschrieben.

Zur benutzerfreundlichen Steuerung der VISART-Werkzeuge und -Adapter-Programme unter UNIX wurde für das X-Window-System eine Benutzeroberfläche mit dem Werkzeug Tcl/Tk [4] entwickelt, die im folgenden vorgestellt wird. (Unter MVS war eine Benutzeroberfläche in Form von 3270-Bildschirm-Panels und ISPF-CLIST-Prozeduren entwickelt worden—siehe alte Fassung von [1c].) Beim Aufruf der Programme von der Benutzeroberfläche aus werden deren Steuereingaben aus den jeweiligen Einstellungen in den Fenstern erzeugt.

Die von den Programmen erwarteten VISART-Dateien dürfen formatiert oder unformatiert sein und im letzteren Fall einfache oder doppelte Genauigkeit der `REAL`-Daten haben. Wenn sie formatiert, d.h. in ASCII geschrieben sind, sind sie ganz unabhängig von ihrer Herkunft lesbar. Wenn sie unformatiert, d.h. binär sind, sind sie im allgemeinen nur lesbar, wenn sie unter dem gleichen Betriebssystem und mit dem gleichen Processor-Typ gelesen werden, wie diejenigen, mit denen sie geschrieben wurden. Da die VISART-Werkzeuge und -Adapter-Programme in unserer Implementierung in Fortran geschrieben sind, müssen unformatierte VISART-Dateien, die z.B. mit C-Programmen erstellt werden und mit diesen Werkzeugen und Adaptern gelesen werden sollen, auch die interne Darstellung der Fortran-Records nachbilden (mit einer Integer-Konstanten am Anfang und Ende des Records, die die Anzahl der Bytes des Records angibt).

### 4.2 Die grafische Benutzeroberfläche

Die Benutzeroberfläche für UNIX erhält man auf den Workstations, auf denen sie implementiert ist, durch Absetzen des Befehls `visart`.

Die Benutzeroberfläche besteht aus einem Hauptfenster (Bild 4.1) und je einem Unterfenster für die ausgewählten Dienste VISARTUL (Bild 4.2), VISAPTER (Bild 4.3), VISAPTER(0) (Bild 4.4), VISARVOL (Bild 4.5) und VISARTOP (Bild 4.8). In einigen der Unterfenster werden bei der Einstellung der Steuerparameter weitere Fenster geöffnet. Bei der Ausführung der Programme werden im Fehlerfall oder—bei VISAPTER und VISARVOL—im Anschluß an die Ausführung des Programms Dialogfenster angeboten, um über die Fortführung des Programms oder—bei VISAPTER und VISARVOL—über den Aufruf der Visualisierungssysteme (Bild 4.6 und 4.7) zu entscheiden.

Jedes Fenster (außer den Dialogfenstern) enthält am oberen Rand eine Leiste von Tasten, mit denen das Fenster geschlossen („Ende“), eine Anleitung abgefragt („Hilfe“) oder ggf. das Programm gestartet werden kann („Ausführung“) oder die Parameter auf ihre Grundstellung gesetzt („Grundstellung“) werden können. Im Hauptfenster gibt es ferner Tasten zur Auswahl der Sprache für die Beschriftung in den Fenstern und für die Texte und Nachrichten („Sprache“) und zum Abfragen der gespeicherten Rundschreiben über den aktuellen Stand der Implementierung („Info“).

In den Unterfenstern folgen unter der Tastenleiste die Anzeigefelder und Kästchen zur Einstellung der zu lesenden VISART-Datei (ggf. mit Fortsetzungs-Files) und—bei VISAPTER und VISARVOL—der zu erstellenden Datei(en) mit der Eingabe für die Visualisierungssysteme bzw.—bei VISARTOP—der zu erstellenden VISART-Datei (ggf. mit Fortsetzungs-Files). Die Dateien lassen sich aus den Anzeigefeldern für die Verzeichnisse und Dateien auswählen oder in den darunterstehenden Kästchen explizit eingeben. Bei Fortsetzungs-Files wird die Auswahl bzw. Eingabe für jeden File wiederholt.

Die weiteren Tasten und Kästchen der Unterfenster werden in den Abschnitten zu den einzelnen Diensten erklärt.

In allen Fenstern findet sich am unteren Rand ein Textfeld, in das Hilfetexte oder das Ausführungsprotokoll bzw. Fehlernachrichten des jeweiligen Programms geschrieben werden. Bei VISARTUL und VISAPTER(0) wird dort auch die aus der Datei ausgewählte Information dargestellt.

Die Bedienung der Benutzeroberfläche geschieht mit der Maus, in der Regel durch einmaliges Anklicken der Tasten im Fenster mit der linken Maustaste (im folgenden gleichbedeutend mit „Drücken“ der Tasten) bzw. durch Doppelklicken in den Anzeigefeldern. Die Kästchen im Fenster werden mit der Tastatur ausgefüllt. Die Funktion aller Tasten, Anzeigefelder, Kästchen ist in Hilfetexten dokumentiert, die durch Anklicken des jeweiligen „Widgets“ mit der rechten Maustaste im Textfeld angezeigt werden.

In den Bildern 4.1 bis 4.8 sind jeweils die Anleitungen zu den Fenstern abgebildet, die man durch Anklicken der Taste „Hilfe“ im Textfeld angezeigt bekommt.

**Bild 4.1: VISART-Hauptfenster**

### 4.3 VISARTUL: Inspektion von VISART-Dateien

Das Programm VISARTUL ist ein Werkzeug zum „Durchblättern“ und Syntax-Prüfen von VISART-Dateien. Die mittels des (einzigen) Eingabeparameters (`LEVEL`) ausgewählten Kopf- und/oder Rumpf-Pakete der VISART-Datei werden im gewählten Detail sequentiell dargestellt.

Bild 4.2 zeigt das VISARTUL-Fenster der grafischen Benutzeroberfläche.

Die Spalten des Tastenfeldes bestimmen die darzustellenden Pakete der VISART-Datei, die Zeilen bestimmen das Detail, mit dem die Gruppen dieser Pakete dargestellt werden. Die Taste „Kompaktinformation“ in der Spalte „Kopf/alle Rumpf“ bewirkt die Ausgabe einer kompakten und kommentierten Beschreibung des ganzen Dateiinhalts; die Taste „, zusaetzl. Zykluskennsaetze“ in derselben Spalte bewirkt die Ausgabe dieser Sätze (und der Beschreibungs- und Geometriesätze), aus denen die Problemzeitpunkte und -zyklen aller vorhandenen Rumpf-Pakete entnommen werden können. Alle Ausgaben erscheinen im Textfeld des Fensters.

Im Anhang A.1 ist ein Beispiel eines mit VISARTUL im vollen Detail dargestellten Kopf- und Rumpf-Pakets abgedruckt. Im Anhang A.2 ist ein Beispiel der von VISARTUL gelieferten Kompaktinformation der gleichen Datei abgedruckt.

Das Programm VISARTUL läßt sich auch als Muster eines Programmteils zum Lesen einer VISART-Datei verwenden. Zu diesem Zweck sind die hierfür relevanten Programmzeilen aus VISARTUL im Anhang A.3 abgedruckt.

**Bild 4.2: VISARTUL-Fenster**

### 4.4 VISAPTER: Visualisierung von Größen

VISAPTER ist ein umfangreiches Adapter-Programm mit zahlreichen Eingabeparametern, das VISART-Dateien in die Dateiformate umsetzt, die von den kommerziellen Visualisierungssystemen (siehe 1.1.3) benötigt werden oder die mittels den dafür entworfenen Command- oder Macro-Files gelesen werden können. (Zu der Sonderfunktion „Werteausgabe“ siehe 4.5.)

Bild 4.3 zeigt das VISAPTER-Fenster der grafischen Benutzeroberfläche mit typischen Einstellungen.

In der Tastenleiste unter den Anzeigefeldern zur Dateiauswahl wird das Visualisierungssystem gewählt (siehe dazu Tabelle in Abschnitt 1.1.3). Jeder Option ist ein Wert des Eingabeparameters `MODE` zugeordnet (siehe nachstehende Tabellen). Normalerweise wird eine node-based Darstellung des Netzes angenommen, falls nicht die Taste „cell-based“ am rechten Rand der Leiste gedrückt wird. (Cell-based Darstellung ist nicht bei jedem Visualisierungssystem möglich—siehe Tabelle in Abschnitt 1.1.3.) Der Wert von `MODE` ist für cell-based Darstellung negativ.

Im darunterliegenden Tastenfeld wird das effektive Koordinatensystem für die Darstellung ausgewählt, das sich aus den geometrischen Koordinaten des ausgeschnittenen Teilgebiets und/oder ggf. der Problemzeit zusammensetzt. (Nicht jedes dieser Koordinatensysteme ist bei jedem Visualisierungssystem möglich—siehe Tabelle in Abschnitt 1.1.3.) Jedem dieser Darstellungskoordinatensysteme entspricht ein Wert des Eingabeparameters `IZ`, der u.a. in den untenstehenden Tabellen und in der Fehlerdiagnostik auftaucht:.

| | $x, y, z$ | $x, y$ | $x$ | $t$ | $x, t$ | $x, y, t$ |
|---|---|---|---|---|---|---|
| `IZ` | -3 | -2 | -1 | 0 | 1 | 2 |

Rechts von diesem Tastenfeld kann eine der folgenden Optionen für das Koordinatensystem (nur bei regulären Netzen) ausgewählt werden:

- „Umrechnung“ von krummlinigen Koordinaten in kartesische Koordinaten (erforderlich bei allen implementierten Visualisierungssystemen—siehe Tabelle in Abschnitt 1.1.3).
- „Abwicklung“ (anstelle der Umrechnung) von Zylindermantel/Kreislinie auf die kartesischen Koordinaten einer Ebene/Gerade bei Zylinderkoordinaten/Polarkoordinaten.
- „Spiegelung“ von zweidimensionalen $r, z$-Netzen oder -Teilgebieten an der $z$-Achse des Koordinatensystems.
- Bei einigen Visualisierungssystemen kann eine „Null-Koordinate“ gewählt werden (s.u.). Parameter dazu werden in einem auf Tastendruck geöffneten Fenster eingestellt.
- Bei den Darstellungskoordinaten $x, t$ und $x, y, t$ kann ein „Zeitfaktor“ eingestellt werden, mit dem die Problemzeit multipliziert wird, um die Länge der Zeitskala auf die gleiche Größenordnung wie die Länge der geometrischen Skala zu bringen.

In den Kästchen der darunterliegenden Leiste werden die Problemzeitpunkte eingestellt, zu denen die Darstellung bei `IZ` $< 0$ erfolgen soll, oder die für die $t$-Koordinate bei `IZ` $\geq 0$ aus dem Bereich der Problemzeiten ausgeschnitten werden soll:

- durch die Angabe der (maximalen) „Anzahl“ der Rumpf-Pakete;
- falls nicht alle Problemzeitpunkte genommen werden sollen, auch durch Angabe der Anzahl und Grenzen von Problemzeit„intervallen“.

**Bild 4.3: VISAPTER-Fenster**

In der nächsten Leiste mit einem Kästchen und weiteren Tasten werden die „Anzahl“ der darzustellenden Größen bzw. Gruppen und ggf. Besonderheiten derselben eingestellt:

- Die Taste „Oberfläche“ muß gedrückt werden, wenn die unten eingestellte Gruppe(n) die darzustellende Größe(n) allein auf der Hülle eines (defektiven) Netzes beschreibt/beschreiben; Parameter dazu werden in einem durch den Tastendruck geöffneten Fenster eingestellt.
- Die Taste „Überlagerung“ muß gedrückt werden, wenn das/die unten eingestellten Gruppenpaar(e)/tripel/...die darzustellende Größe(n) im Innern und auf der Hülle eines (defektiven) Netzes beschreibt/beschreiben; Parameter dazu werden in einem durch den Tastendruck geöffneten Fenster eingestellt.

In den Kästchen des letzten Feldes werden die Größen bzw. Gruppen eingestellt, und zwar pro Leiste durch Angabe

- der „Gruppen“kennzahl der Größe;
- der „Identifikation“ der Größe (bei den Gruppen 19 ist die Angabe leer oder gleich der Identifikation der Größe in einer beschreibenden Gruppe 9);
- der Identifikation der „Referenz“gruppe 6/16 bei den Subgruppen 7/17 bzw. der Identifikation der Gruppe 19 oder ggf. nur der ersten Zeichen derselben, soweit sie mit der Identifikation der beschreibenden Gruppe 9 übereinstimmen (sonst leer);
- der „Komponenten“nummer, falls auf eine skalare Komponente eines Vektors zugegriffen werden soll;
- der „Anfangs“- und „End“maschenindizes des Teilgebiets bzw. der festzuhaltende Maschenindex für Subnetze, falls ein Teilgebiet bzw. ein Subnetz ausgeschnitten werden soll (bei Gruppen 19 ohne beschreibende Gruppe 9 auch die Position der Größe in der Gruppe).

VISAPTER akzeptiert auch mehr als nur eine VISART-Datei, die aus verschiedenen Code-Läufen, ggf. jeweils mit Fortsetzungs-Files, stammen können. In diesem Fall ist für jede Datei nach der Einstellung der Eingabe (was außer bei der ersten Datei nur im unterhalb der Zeile „Rumpfpakete ...“ stehenden Fensterteil möglich ist) die Taste „$n$ Eingaben“ in der oberen Leiste zu drücken (worauf $n$ auf die Anzahl der bisherigen Eingaben springt).

Eine vollständige Beschreibung der Funktionen findet man in den on-line Hilfetexten zu den einzelnen „Widgets“ der Benutzeroberfläche und im Teil 3 [1c] der VISART-Dokumentation.

Für den Inhalt der von VISAPTER erwarteten VISART-Datei(en) gilt:

VISAPTER entnimmt die Netzkoordinaten bei regulären Netzen der Gruppe 4 bzw. bei irregulären Netzen den Gruppen 5/15, die die Identifikationen `..COORD..` tragen.

Bei Netzen mit Löchern entnimmt VISAPTER die Fehlstellen des Netzes den Gruppen 5/15, die die Identifikation `..DEFCT..` tragen (dort bezeichnen 0 die belegten Maschen, $\neq 0$ die leeren Maschen).

VISAPTER kann bis jetzt noch keine Gruppen 51/151 und 71/171 (über mehrfach unterteilten Netzen) verarbeiten.

Einschränkungen bezüglich der Verträglichkeit der Eingabeparameter mit den Visualisierungs- und Darstellungskoordinatensystemen ergeben sich aus den nachfolgenden Tabellen.

Die folgende Tabelle zeigt die Verträglichkeit einiger Parameter mit den Visualisierungssystemen.

| MODE | | IZ | Film | Null-Ko'te | Spiegelung | Oberfläche | Überlagerung |
|---|---|---|---|---|---|---|---|
| 0 | Werteausgabe | -3...+2 | x | | | | |
| 1 | AVS-FD | -3...+2 | | x | x | x | x |
| 3 | UNIGRAPH | -3...+2 | x | | x | x | x |
| 4 | AVS-UCD, n.b. | -3...-1,+1,+2 | | x | x | x | x |
| −4 | AVS-UCD, c.b. | -3...-1 | | x | x | x | |
| 5 | Tecplot, n.b. | -3...+2 | x | | x | x | x |
| −5 | Tecplot, c.b. | -3...-1 | x | | x | x | |

Erklärungen:

- „n.b.“ bedeutet node-based, „c.b.“ bedeutet cell-based.
- „Film“ bedeutet die Möglichkeit der gleichzeitigen Ausgabe für mehrfache Problemzeiten bei IZ $< 0$.
- „Null-Ko'te“ bedeutet komplementär dazu die Möglichkeit, bei IZ $> 0$ die Werte der Zeitkoordinate in der Ausgabe Null zu setzen, dabei aber über den verbleibenden (geometrischen) Koordinaten mehrfach (entsprechend den genullten Problemzeitpunkten) auszugeben (Anwendung siehe Teil 3 [1c]).

(Mit den Möglichkeiten „Film“ und „Null-Ko'te“ kann eine Folge von Darstellungen über der Geometrie für eine Folge von Problemzeitpunkten ausgegeben werden, die sich für eine Verfilmung eignet.)

Die folgende Tabelle zeigt die Verträglichkeit der Gruppen mit den Visualisierungssystemen und den Darstellungskoordinatensystemen.

| | Gruppe | 5/15 15 | | 5/15 15 | | 5/15 15 | | 7/17 17 | | 16+17 | 19 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Netz | voll | | Teilnetz | | Löcher | | beliebig | | kein | bel. | bel. |
| MODE | IZ | $\neq 0$ | $= 0$ | $\neq 0$ | $= 0$ | $\neq 0$ | $= 0$ | $\neq 0$ | $= 0$ | $< 0$ | $= 0$ | $= 0$ |
| 0 | Werteausgabe | x | x | x | x | x | x | x | x | | x | x |
| 1 | AVS-FD | x | x | | x | | x | | x | | x | x |
| 3 | UNIGRAPH | x[3] | x | | x | x[3] | x | x[3,4] | x | | x | x |
| ±4 | AVS-UCD | x | | x | | x | | x | | | | |
| ±5 | Tecplot | x | x | x | x | x | x | x[4] | x | x | x | x |

[3] nur für reguläre Netze und kartesische Koordinaten
[4] durch Expansion auf Gruppen 5/15 mit Löchern

### 4.5 VISAPTER(0): Werteausgabe von Größen

Das Programm VISAPTER (siehe 4.4) enthält als Sonderfunktion (Eingabeparameter `MODE` = 0) die Ausgabe von Zahlenwerten der Größen auf die Protokolldatei. Wie bei den anderen Modes von VISAPTER können für die ausgewählten Größen Teilgebieten und Problemzeiten spezifiziert werden, und zwar hier für alle Darstellungsdimensionen, d.h. auch für einzelne Maschen und/oder Zeitpunkte. Die Zahlenwerte der Größen werden zusammen mit ihren Koordinaten angezeigt, wahlweise ohne oder mit Umrechnung von krummlinigen auf kartesische Koordinaten.

Bild 4.4 zeigt das VISAPTER(0)-Fenster der grafischen Benutzeroberfläche mit typischen Einstellungen.

Die Funktion der Tasten, Kästchen usw. ist identisch mit der der Benutzeroberfläche von VISAPTER (siehe 4.4). Eine Ausnahme ist das Tastenfeld für die Darstellungskoordinaten, wo nur zwischen den Kombinationen von Geometrie (ohne deren Dimension) und Zeit gewählt werden muß (entsprechend dem Vorzeichen des Eingabeparameters `IZ`; der zutreffende Betrag von `IZ` wird vom Programm anhand der Eingaben im nachstehenden Teil des Fensters automatisch bestimmt).

Die Ausgabe der Zahlenwerte erscheint im Textfeld des Fensters.

Eine vollständige Beschreibung der Funktionen findet man in den on-line Hilfetexten zu den einzelnen „Widgets“ der Benutzeroberfläche und im Teil 3 [1c] der VISART-Dokumentation.

Für den Inhalt der von VISAPTER(0) erwarteten VISART-Datei(en) gilt das in 4.4 Gesagte.

**Bild 4.4: VISAPTER(0)-Fenster**

### 4.6 VISARVOL: Darstellung von Volumenanteilen

Das Programm VISARVOL ähnelt dem Programm VISAPTER in vielem, ist aber spezialisiert auf die Darstellung der Volumenanteile verschiedener Materialien über den Maschen 2dimensionaler Netze/Subnetze/Teilgebiete. Wie VISAPTER setzt VISARVOL VISART-Dateien in die Dateiformate um, die von den kommerziellen Visualisierungssystemen (siehe Tabelle in Abschnitt 1.1.3)[1] benötigt werden oder die mittels den dafür entworfenen Command- oder Macro-Files gelesen werden können.

Bild 4.5 zeigt das VISARVOL-Fenster der grafischen Benutzeroberfläche mit typischen Einstellungen.

In der Tastenleiste unter den Anzeigefeldern zur Dateiauswahl wird das Visualisierungssystem gewählt.

In den Kästchen der darunterliegenden Leiste werden die Problemzeitpunkte eingestellt, zu denen die Darstellung erfolgen soll:

- durch die Angabe der (maximalen) „Anzahl“ der Rumpf-Pakete;
- falls nicht alle Problemzeitpunkte genommen werden sollen, auch durch Angabe der Anzahl und Grenzen von Problemzeit„intervallen“.

In der nächsten Zeile wird in einem Kästchen die „Anzahl“ der zusammen zu visualisierenden Größen (Volumenanteile von Materialien) eingestellt.

In den Kästchen der nächsten Leiste werden die allen Größen gemeinsamen Parameter eingestellt, und zwar

- die „Gruppen“kennzahl (5/15 oder 7/17);
- die Identifikation der „Referenz“gruppe 6/16 bei den Subgruppen 7/17 (sonst leer);
- die „Anfangs“- und „End“maschenindizes des Teilgebiets bzw. der festzuhaltende Maschenindex für Subnetze, falls ein (2dimensionales) Teilgebiet oder Subnetz ausgeschnitten werden muß.

In den Kästchen und mit den Menütasten des letzten Feldes werden die Größen (Volumenanteile von Materialien) und deren Darstellungsart eingestellt, und zwar durch Angabe

- der Identifikation der Größe („Material“);
- der „Phase“ bei der Darstellung des Volumenanteils (verteilt oder getrennt);
- der Symbol„zeichen“ zur Darstellung des Materials (Einfärbung, Schraffuren, Tropfen, Blasen);
- der Symbol„farbe“ zur Darstellung des Materials;
- der „Abstufung“ von Farbsättigung, Schraffurabstand, Tropfen- oder Blasengröße.

(Die letzten 4 Spalten dieses Feldes werden mit Pull-Down-Menüs eingestellt.)

VISARVOL akzeptiert auch (wie VISAPTER) mehr als nur eine VISART-Datei, die aus verschiedenen Code-Läufen stammen können (siehe 4.4).

Eine vollständige Beschreibung der Funktionen findet man in den on-line Hilfetexten zu den einzelnen „Widgets“ der Benutzeroberfläche und im Teil 5 [1e] der VISART-Dokumentation.

Für den Inhalt der von VISARVOL erwarteten VISART-Datei(en) gilt das gleiche wie bei VISAPTER (siehe 4.4).

[1] bis jetzt nur Tecplot

**Bild 4.5: VISARVOL-Fenster**

### 4.7 Aufruf der Visualisierungssysteme

Die Programme VISAPTER und VISARVOL erzeugen eine Eingabedatei[1] (formatiert, d.h. in ASCII Character-Darstellung) für das gewählte Visualisierungssystem. Für das Format dieser Dateien gilt:

- Für AVS ist das Format der Eingabedatei(en) festgelegt. VISAPTER schreibt sie in diesem Format.
- Für UNIGRAPH kann das Format der Eingabedatei beliebig sein, da die Datei im System noch manipuliert werden kann. VISAPTER schreibt sie in einem speziell ausgelegten Format, das alle für den Bildaufbau und die Bildbeschriftung benötigte Information enthält. Parallel dazu wird mit der Benutzeroberfläche ein Command-File zur Verfügung gestellt, der dieses Format beim UNIGRAPH-Aufruf verarbeitet und daraus eine dem Fall angemessene Standarddarstellung erzeugt.
- Für Tecplot besteht die Eingabedatei aus fakultativen Abschnitten, die ihrerseits ein festes Format haben. VISAPTER schreibt sie in einem von drei speziell für die jeweiligen Fälle ausgelegten Kombinationen. Parallel dazu werden mit der Benutzeroberfläche Macro-Files zur Verfügung gestellt, die die Datei beim Tecplot-Aufruf einlesen und die jeweilige Standarddarstellung erzeugen. (Für VISARVOL gilt ähnliches.)
- Bei UNIGRAPH und Tecplot enthalten die Dateien auch den Problemtitel, der—ebenso wie die Identifikation der Größe(n), sowie ggf. die Subnetzebene(n) und die Problemzeit—im Bild mit dargestellt wird.

Um den Benutzern die Arbeit zu erleichtern, wird unter dem VISAPTER-Fenster der Benutzeroberfläche nach der Ausführung von VISAPTER ein Dialogfenster eingeblendet, in dem der Aufruf des gewählten Visualisierungssystems angeboten wird, wahlweise gesteuert über einen der Command-/Macro-Files für die soeben erzeugte Eingabedatei. Hierbei läuft dann das Einlesen der Datei und der Aufbau des Bildes automatisch ab, bis zu einer dem Problem angepaßten Standarddarstellung. Anschließend kann der Benutzer wieder die Regie übernehmen. Für VISARVOL gilt ähnliches.

Bild 4.6 zeigt das Dialogfenster für den UNIGRAPH-Aufruf aus dem VISAPTER-Fenster; Bild 4.7 zeigt das Dialogfenster für den Tecplot-Aufruf aus dem VISAPTER-Fenster.

Mit der Taste „Visualisierung“ im VISAPTER- und VISARVOL-Fenster kann das Dialogfenster jederzeit auch ohne unmittelbar vorhergehende Programmausführung geöffnet werden, um das gewählte Visualisierungssystem für eine bereits vorhandene Eingabedatei aufzurufen.

[1] für AVS-FD zwei Eingabedateien

**Bild 4.6: UNIGRAPH-Dialogfenster**

**Bild 4.7: Tecplot-Dialogfenster**

### 4.8 VISARTOP: Manipulation von VISART-Dateien

Das Programm VISARTOP ist ein Werkzeug zum Bearbeiten von VISART-Dateien und zum Ableiten neuer Größen aus den in der Datei enthaltenen Größen. VISARTOP erzeugt aus der gelesenen eine neue VISART-Datei, die die gewünschten Änderungen oder Erweiterungen enthält.

Bild 4.8 zeigt das VISARTOP-Fenster der grafischen Benutzeroberfläche.

In der Tastenleiste unter den Anzeigefeldern zur Dateiauswahl wird ausgewählt,

- ob eine formatierte VISART-Dateien in eine unformatierte (oder umgekehrt) umgewandelt werden soll, oder ob das Format der gelesenen Datei übernommen werden soll;
- ob Fortsetzungs-Files (aus Restart-Läufen) mit dem File aus dem Erstlauf vereinigt werden sollen, oder ob auch die neue Datei ggf. aus Fortsetzungsfiles bestehen soll.

In den Kästchen der nächsten Leiste werden ggf. die Rumpf-Pakete spezifiziert, die in die neue Datei übernommen werden sollen (bei fehlenden Angaben werden alle übernommen). Dies geschieht durch die Angabe

- eines Problemzyklustaktes (z.B. 3: jedes 3. Rumpf-Paket);
- von (einem oder mehreren) Problemzyklusintervallen (z.B. 0 und 17);
- von (einem oder mehreren) Problemzeitintervallen (z.B. 0. und 0.02);
- von (einer oder mehreren) Zyklusidentifikationen (z.B. CYCLFINI).

Nun folgen zwei Bereiche mit Tasten, Kästchen und Feldern, deren erster sich auf das Kopf-Paket und deren zweiter sich auf die Rumpf-Pakete der Dateien bezieht. Abgesehen davon sind sie identisch.

- Mit dem oberen Tastenpaar und in den Kästchen der darunterstehenden Leiste werden die Gruppen im Kopf-Paket bzw. in den Rumpf-Paketen spezifiziert, die für die neue Datei ausgewählt oder verworfen werden sollen. (Wird die Taste „Nicht zu übernehmende“ gedrückt, und bleiben die Kästchen leer, so werden alle Gruppen übernommen; wird die Taste „Zu übernehmende“ gedrückt, und bleiben die Kästchen leer, so werden keine Gruppen übernommen.)
- Im Textfeld unter der Taste „ja“ werden Anweisungen (s.u.) zum Ableiten neuer Größen spezifiziert, die als VISART-Gruppen auf die neue Datei geschrieben werden sollen. Beim Drücken der Taste „ja“ wird ein Fenster geöffnet, in dem mehr Zeilen für das Textfeld zur Verfügung stehen, und im dem das Textfeld auch aus einer Datei gefüllt bzw. in eine Datei übertragen werden kann.

Eine vollständige Beschreibung der Funktionen findet man in den on-line Hilfetexten zu den einzelnen „Widgets“ der Benutzeroberfläche und im Teil 4 [1d] der VISART-Dokumentation.

Das Ableiten neuer Größen aus in der Datei vorhandenen Größen, durch arithmetische, logische und andere Operationen und Funktionen (siehe die beiden nachfolgenden Tabellen), ist der wichtigste Zweck von VISARTOP. Die Steuerung erfolgt durch die Eingabe einfacher Anweisungen, die lediglich die Identifikationen der Größen benützen. Aus der Vielzahl von Möglichkeiten können hier nur einige Beispiele gezeigt werden; im übrigen muß auf den Teil 4 [1d] der VISART-Dokumentation verwiesen werden.

**Bild 4.8: VISARTOP-Fenster**

**Beispiele für Anweisungen**

(Wenn nichts anderes gesagt ist, beziehen sich die Anweisungen auf Gruppen aus den Rumpf-Paketen der Datei.)

In den folgenden Anweisungen sind die links vom Gleichheitszeichen stehenden Größen temporär und werden nicht auf die Ausgabedatei geschrieben, wenn ihre Identifikation mit einem Dollarzeichen beginnt; andernfalls werden sie als neue Größen auf die Ausgabedatei geschrieben. Die temporären und neuen Größen erhalten die angegebene Identifikation und die gleiche Gruppenkennzahl, wie die rechts stehenden Größen.

Die in den Anweisungen rechts vom Gleichheitszeichen stehenden Ausdrücke enthalten als Operanden Konstanten, Identifikationen von Größen aus der Eingabedatei oder von bereits in vorhergehenden Anweisungen definierten temporären Größen, oder Funktionsaufrufe. Die (ggf. qualifizierten—s.u.) Größen in den Ausdrücken müssen in der Gruppenkennzahl, in der Länge und in der Vektordimension zueinander passen.

Die Identifikationen der Größen können mit Qualifikatoren versehen sein, die z.B. eine Vektorkomponente bezeichnen (@ mit nachfolgender Komponentennummer), Werte über einem Subnetz oder in einer Einzelmasche auswählen (durch Angabe der festgehaltenen Indizes in geschweiften Klammern), eine integrale Größe spezifizieren (durch Angabe ihrer Position in der Gruppe 19 in geschweiften Klammern), oder Lokationen auf Maschenhüllen den Maschenmitten (oder umgekehrt) zuordnen (durch Verschiebungspfeile in geschweiften Klammern).

Beispiel 1: Einfache Kombination von Größen (Gruppen 15) durch Addition bzw. Bildung des Maximums zweier Größen:

```
RBR1 = RSBR1 + RSBR3 + RLBR1 + RLBR6 + RGBR1;

RGBRX = (RGBR2 > RGBR3) ?  RGBR2 :  RGBR3;
```

(Im Ausdruck auf der rechten Seite der zweiten Anweisung wird eine Konvention aus der Programmiersprache C zugrunde gelegt: Wenn die Bedingung (1. Operand) wahr ist, erhält der Ausdruck den Wert des 2. Operanden, andernfalls den Wert des 3. Operanden.)

Beispiel 2: Ausschnitt einer Größe (Gruppe 15) über einem zweidimensionalen Subnetz des dreidimensionalen Netzes:

```
PS = P{0,1,0} ;
```

(Die 2. Koordinate des Netzes wird auf dem Maschenindex 1 festgehalten.)

Beispiel 3: Betragsbildung aus den Komponenten einer Vektorgröße (Gruppe 15) über einem zweidimensionalen Netz (hier der Geschwindigkeit in der Maschenmitte):

```
ABSVELL = %W ( %Q VELL@1 + %Q VELL@2 ) ;
```

(`%Q` und `%W` sind monadische Operatoren; siehe Tabelle.)

Beispiel 4: Aufbau von Vektorgrößen aus skalaren Größen (Gruppe 15) über einem zweidimensionalen Netz (hier der Geschwindigkeit in der Maschenmitte aus den Geschwindigkeitskomponenten auf den Maschenhüllen):

```
$VELL@1 = 0.5 * ( UL{<,} + UL{>,} ) ;
$VELL@2 = 0.5 * ( VL{,<} + VL{,>} ) ;
VELL = $VELL ;
```

(Die Verschiebungspfeile vor dem Komma beziehen sich auf die 1. Koordinatenrichtung, die nach dem Komma auf die 2. Koordinatenrichtung.)

Beispiel 5: Wichtung von Geschwindigkeitskomponenten auf den Maschenhüllen mit den Volumenfraktionen der stromaufwärtsliegenden Maschen eines zweidimensionalen Netzes (Gruppen 15):

```
- $ALL{>,} = AL ;
- $ALR{<,} = AL ;
- UAL = ( UL > 0.  )  ?  ( UL * $ALL ) : ( UL * $ALR ) ;
- $ALB{,>} = AL ;
- $ALT{,<} = AL ;
- VAL = ( VL > 0.  )  ?  ( VL * $ALB ) : ( VL * $ALT ) ;
```

(Die Verschiebungspfeile vor dem Komma beziehen sich auf die 1. Koordinatenrichtung, die nach dem Komma auf die 2. Koordinatenrichtung. `UAL` und `VAL` sind wie `UL` bzw. `VL` den Maschenhüllen zugeordnet.)

Beispiel 6: Integration einer Größe (Gruppe 15) über dem Netz (hier einer Dichte multipliziert mit dem Maschenvolumen):

- *Anweisung für das Kopf-Paket:*
  `$VOL = VOLUMES ;`
- *Anweisung für die Rumpf-Pakete:*
  `$MASSL3 = #05 ( RLBR3 * $VOL ) ;`

(`#05` steht für den Aufruf der Funktion für die Integration (mit dem Argument im nachfolgenden Klammerpaar; siehe Tabelle). `$MASSL3` ist keine Gruppe, sondern ein „Single“, der in weiteren Anweisungen wie eine Konstante benutzt werden kann oder als integrale Größe in einer Gruppe 19 ausgegeben werden kann; siehe Beispiel 7.)

Beispiel 7: Erzeugung einer Gruppe 19 (hier mit 57 Größen, die mit 0. initialisiert werden); Zuweisung des im Beispiel 6 erzeugten Singles an die Position 16, und Ausgabe der Gruppe:

```
- $INTGRL = #02 ( 57 , 0.  )  ;
- ........
- $INTGRL{16} = $MASSL3 ;
- ........
- INTGRL = $INTGRL ;
```

(`#02` steht für den Aufruf der Funktion für die Erzeugung einer Gruppe 19 (mit den Argumenten im nachfolgenden Klammerpaar; siehe Tabelle).)

**Operationen und Operatoren**

| Operator[3] | | Operation | Typ des Resultats zum Operanden-Typ | | | | |
|---|---|---|---|---|---|---|---|
| | | *Monadische Operationen* | R | | I | L | C |
| `+` | v | keine[1] | R | - | I | - | - |
| `-` | v | Vorzeichenumkehr[1] | R | - | I | - | - |
| `%M` | v | Vorzeichenumkehr | R | - | I | - | - |
| `%K` | v | Kehrwert | R | - | - | - | - |
| `%A` | v | Betrag | R | - | I | - | - |
| `%S` | v | Signum | R | - | I | - | - |
| `%Q` | v | Quadrat | R | - | I | - | - |
| `%W` | v | Wurzel | R | - | - | - | - |
| `%E` | v | Exponentialfunktion | R | - | - | - | - |
| `%L` | v | Natürlicher Logarithmus | R | - | - | - | - |
| `%R` | v | Umwandlung in REAL | - | - | R | - | - |
| `%I` | v | Umwand. in INTEGER | I | - | - | I | - |
| `%G` | v | Umwand. in LOGICAL[2] | L | - | L | - | - |
| `%N` | v | Negation | - | - | - | L | - |
| | | *Dyadische Operationen* | R,R | R,I | I,I | L,L | C,C |
| | | *arithmetische und logische* | | | | | |
| `+` | z | Addition | R | R | I | - | - |
| `-` | z | Subtraktion | R | R | I | - | - |
| `*` | z | Multiplikation | R | R | I | - | - |
| `/` | z | Division | R | R | I | - | - |
| `&` | z | logisches Und | - | - | - | L | - |
| `\|` | z | logisches Oder | - | - | - | L | - |
| | | *relationale* | | | | | |
| $<$ | z | kleiner | L | - | L | - | - |
| $<=$ | z | kleiner oder gleich | L | - | L | - | - |
| $>$ | z | größer | L | - | L | - | - |
| $>=$ | z | größer oder gleich | L | - | L | - | - |
| $==$ | z | identisch | L | - | L | - | L |
| $!=$ | z | nicht identisch | L | - | L | - | L |
| | | *Triadische Operationen* | R | | I | L | C |
| `?` | n | Bedingung abfragen | - | - | - | L | - |
| `:` | z | bedingte Zuweisung | R | - | I | L | C |

1 2 3

[1] nur am Anfang eines Ausdrucks zulässig
[2] Operand $\neq 0$ gibt Resultat `.true.`
[3] v: vor dem Operand; n: nach dem Operand; z: zwischen den Operanden

**Funktionen**

Funktionen werden durch

`#`*nn* `(`Argumente`)`

aufgerufen, wo *nn* die Nummer der Funktion in der folgenden Tabelle ist:

`01`: Erzeugung einer Gruppe 5 oder 15 mit konstantem Wert (Argument 1) in allen Maschen.

`02`: Erzeugung einer Gruppe 9 oder 19 mit vorgegebener Anzahl (Argument 1) von Elementen und konstantem Wert (Argument 2) in allen Elementen.

`03`: Erzeugung eines Singles mit dem Wert des Maximums (ggf. jeder Komponente) der Maschenwerte einer beliebigen Gruppe (Argument 1).

`04`: Erzeugung eines Singles mit dem Wert des Minimums (ggf. jeder Komponente) der Maschenwerte einer beliebigen Gruppe (Argument 1).

`05`: Erzeugung eines Singles mit dem Wert der Summe (ggf. jeder Komponente) der Maschenwerte einer beliebigen Gruppe (Argument 1).

`06`: Expansion einer Subgruppe 7 oder 17 (Argument 1) auf eine Gruppe 5 bzw. 15 gemäß den Koordinaten in der zugehörigen Beschreibungsgruppe 6 bzw. 16; mit Default-Werten (Argument 2) der Gruppe 5 bzw. 15.

`07`: Kompression einer Gruppe 5 bzw. 15 (Argument 1) in eine Subgruppe 7 oder 17 gemäß den Koordinaten in der zugehörigen Beschreibungsgruppe 6 bzw. 16.

`08`: Erzeugung eines Singles mit dem Wert der Position von `.true.` in einer beliebigen Gruppe (Argument 1); bei mehrfachem Vorkommen von `.true.` in der Gruppe: mit der Position des *n*-ten Auftretens (Argument 2).

`09`: Erzeugung eines Singles mit dem Wert einer beliebigen Gruppe (Argument 1) an einer gegebenen Position (Argument 2)

`10`: Erzeugung einer Gruppe 9/19 mit den Werten der Positionen von `.true.` in einer beliebigen Gruppe (Argument 1).

`11`: Erzeugung einer Gruppe 9/19 mit den Werten einer beliebigen Gruppe (Argument 1) an den Positionen, die durch eine Gruppe 9/19 (Argument 2) gegeben sind.

`12`: Erzeugung einer skalaren Gruppe 5/15 auf den Lokationen -88 mit konstanten Werten auf der inneren Netzhülle (Argument 1) und auf der äußeren Netzhülle (Argument 2) in den Koordinatenrichtungen.

`13`: Erzeugung einer Subgruppe 7 oder 17 mit konstantem Wert (Argument 1) in den gemäß der zugehörigen Beschreibungsgruppe 6 bzw. 16 definierten Koordinaten.

`14`: Erzeugung einer Gruppe (Kennzahl wie Argument 1) mit den Positionen der Werte einer beliebigen Gruppe (Argument 1) in sortierter Reihenfolge (numerisch bzw. alphabetisch).

`15`: Erzeugung einer Gruppe (Kennzahl wie Argument 1) mit den Werten einer beliebigen Gruppe (Argument 1), sortiert nach den Positionen in einer anderen Gruppe (Argument 2, Kennzahl wie Argument 1).

`16`: Expansion einer Subgruppe 7 oder 17 (Argument 1) auf eine Gruppe 5 bzw. 15 gemäß den Koordinaten in der Beschreibungssubgruppe 7/17 (Argument 3) mit Default-Werten (Argument 2) der Gruppe 5 bzw. 15.

`17`: Kompression einer Gruppe 5 bzw. 15 (Argument 1) in eine Subgruppe 7 oder 17 gemäß den Koordinaten in der Beschreibungssubgruppe 7/17 (Argument 2).

## Literatur


[1b] S. Kleinheins:
Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen
Teil 2: VISART — ein standardisiertes Postprocessor-Dateiformat
FZKA 5996, 1997

[1c] S. Kleinheins:
Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen
Teil 3: VISAPTER — ein Programm zum Anschluß einiger Visualisierungssysteme an VISART-Dateien
FZKA 5997, 1997

[1d] S. Kleinheins:
Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen
Teil 4: VISARTOP — ein Programm zum Bearbeiten von VISART-Dateien
FZKA 5998, 1997

[1e] S. Kleinheins:
Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen
Teil 5: VISARVOL — ein Programm zur Visualisierung von Volumenanteilen aus VISART-Dateien
FZKA 5999, 1997

[2a] W.R. Bohl, L.B. Luck:
SIMMER-II: A Computer Program for LMFBR Disrupted Core Analysis
LA-11415-MS, LANL, June 1990

[2b] W.R. Bohl et al.:
AFDM: An Advanced Fluid-Dynamics Model
LA-11692-MS, LANL, September 1990

[2c] Sa. Kondo et al.:
SIMMER-III: A Computer Program for LMFBR Core Disruptive Accident Analysis
Version 2.A: User's Manual
PNC ZN9410 96-207, June 1996

[2d] C.-D. Munz, S. Hirmer:
Unveröffentlichter Bericht, April 1992

[2e] N.I. Kolev:
IVA3: Computer Code for Modelling of Transient Three Dimensional Three Phase Flow in Complicated Geometry
Program Documentation: Input Description
KfK 4950, Dezember 1991

[2f] H. Borgwaldt, W. Baumann, G. Willerding:
FLUTAN
Input Specification
KfK 5010, Mai 1992

[2g] E. Halter et al.:
A Concept for the Numerical Solution of the Maxwell-Vlasov System
FZKA 5654, Oktober 1995

[2h] C.-D. Munz et al.:
Numerische Simulation von Hochstrom-Ionendioden und Gyrotron-Oszillatoren
FZK Nachrichten 28, 2-3 (1996), 233-248

[2i] J.R. Travis et al.:
GASFLOW-II: A Three-Dimensional Finite-Volume Fluid-Dynamics Code for Calculating the Transport, Mixing, and Combustion of Flammable Gases and Aerosols in Geometrically Complex Domains.
FZKA-Bericht in Vorbereitung

[3a] AVS User's Guide
Release 4, May 1992
Advanced Visual Systems Inc., Waltham MA, USA, 1992

[3c] AVS/Uniras
UNIGRAPH +2000
Release 6.4a User's Guide
Advanced Visual Systems Inc., Waltham MA, USA, 1994

[3d] Tecplot (Interactive Data Visualization for Scientists & Engineers)
Version 6 User's Manual
Amtec Engineering, Inc., Bellevue WA, USA, 1994

[4] J.K. Ousterhout:
Tcl und Tk
(Entwicklung grafischer Benutzerschnittstellen für das X Window System)
Addison-Wesley, Bonn 1995

# Anhang

## A. Beispiele zu VISART-Dateien

### A.1 Beispiel einer im vollen Detail ausgedruckten VISART-Datei

Die nachstehende Liste veranschaulicht den Aufbau einer Postprocessor-Datei im VISART-Format anhand von Ausschnitten eines (mit VISARTUL, `LEVEL` = 4, ausgedruckten) realen Beispiels. Es wurde mit dem SIMMER-III-Code [2c] erzeugt, der auf einem regulären, vollen Netz mit achsensymmetrischer Kreiszylindergeometrie beruht. In diesem Ausdruck folgen die Zahlen und Zeichenketten so aufeinander, wie sie in der Datei stehen; lediglich die Trennlinien für Kopf und Rumpf, die Zeilen vor der Kopf-Trennlinie und die Sterne als Beginn der Datensätze wurden von VISARTUL zugefügt.

Neben den obligatorischen Gruppen mit den in 3.2 definierten Identifikationen kommen in der Datei die folgenden Gruppen vor:

im Kopf-Paket:
- `WIDTHS` (Gruppenkennzahl 5, Dimension 2) enthält maschenweise die Weiten der Maschen in den Koordinatenrichtungen;
- `VOLUMES` (Gruppenkennzahl 5, Dimension 0) enthält maschenweise die Volumina der Maschen;
- `BOUNDARY` und `INITIAL` (Gruppenkennzahl 9) enthalten Rand- bzw. Anfangswerte des Problems;
- `CELL_NM` (Gruppenkennzahl 9) enthält die Identifikationen der in den Rumpf-Paketen stehenden Netzgrößen;
- `INTGRLNM` (Gruppenkennzahl 9) definiert die Namen der Integralgrößen, deren Werte in den Rumpf-Paketen unter `INTGRLVL` (Gruppenkennzahl 19) (in derselben Reihenfolge, komponentenweise verschachtelt) erscheinen;
- `ALLTIMNM` (Gruppenkennzahl 9) enthält die Identifikationen der in den Rumpf-Paketen stehenden Zeitfunktionsgrößen.

in den Rumpf-Paketen:
- `VEL___2` (Gruppenkennzahl 15, Dimension 2) enthält maschenweise die Geschwindigkeitskomponenten des Feldes 2;
- `ALPLK_3` (Gruppenkennzahl 15, Dimension 0) enthält maschenweise die Volumenanteile der Materialkomponente 3;
- `INTGRLVL` (Gruppenkennzahl 19) enthält die Werte der Integralgrößen, deren Namen im Kopf-Paket unter `INTGRLNM` (Gruppenkennzahl 9) (in derselben Reihenfolge, komponentenweise verschachtelt) erscheinen;
- `TIMEALL` (Gruppenkennzahl 20) enthält die Problemzeitpunkte für die nachfolgenden Gruppen 21;
- `ALPLK_3T` (Gruppenkennzahl 21, Netzgröße) enthält den Volumenanteil der Materialkomponente 3 in der Masche 1,5 zu den in der Gruppe 20 definierten Problemzeitpunkten;
- `MASL_6T` (Gruppenkennzahl 21, Integralgröße) enthält die Masse der flüssigen Materialkomponente 6 im Netz zu den in der Gruppe 20 definierten Problemzeitpunkten.

```
VISARTUL  -  VISART-RELEASE 1.22    -  OPTIONS: UNIX ON, DBLPP OFF, SCREEN
LEVEL= 4

UNFORMATTIERTE DATEI, 1-FACHE REAL-WORTLAENGE, RELEASE 1.22

----------------------KOPF-----------------------------------------------

  1         0  SIMMER-I  II 2.A    INR308    96-11-06  10:19:32

  2         0  inrrisc6            kleinh1   96/11/ 7  11.10.55

  3         2
LITTLE WORK ENERGY PROBLEM  EURO-PNC FOR S-III VER.2.A


  4         3  GEOMETRY         2         1       220
            4        13         0        33  0.000E+00  0.000E+00  0.000E+00
            *  0.0000E+00  9.3500E-01  1.8700E+00  2.8050E+00
            *  0.0000E+00  9.1440E-01  1.8288E+00  2.7432E+00  3.6576E+00
               4.5720E+00  5.4864E+00  6.4008E+00  7.3152E+00  8.2296E+00
               9.1440E+00  1.0058E+01  1.0973E+01
  5         3  WIDTHS          36         2          1
      0     0    0    0    0    0    0    0         12          0
            *  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01
               9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01
               9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01
               9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01
               9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01
               9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01
               9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01  9.3500E-01
               9.3500E-01
            *  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01
               9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01
               9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01
               9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01
               9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01
               9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01
               9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01  9.1440E-01
               9.1440E-01
  5         2  VOLUMES          36         0          1
      0     0    0    0    0    0    0    0         12          0
            *  2.5114E+00  7.5341E+00  1.2557E+01  2.5114E+00  7.5341E+00
               1.2557E+01  2.5114E+00  7.5341E+00  1.2557E+01  2.5114E+00
               7.5341E+00  1.2557E+01  2.5114E+00  7.5341E+00  1.2557E+01
               2.5114E+00  7.5341E+00  1.2557E+01  2.5114E+00  7.5341E+00
               1.2557E+01  2.5114E+00  7.5341E+00  1.2557E+01  2.5114E+00
               7.5341E+00  1.2557E+01  2.5114E+00  7.5341E+00  1.2557E+01
               2.5114E+00  7.5341E+00  1.2557E+01  2.5114E+00  7.5341E+00
```

```
              1.2557E+01
 9         1  BOUNDARY         1         0         0
           *          0
 9         1  INITIAL          1         0         1
           *  1.0000E-03
 9         1  CELL  NM         2         0         2
           *    VEL   2     ALPLK 3
 9         1  INTGRLNM        46         0         2
           *    MASSLO1     MASSLO2     MASSLO3     MASSLO4     MASSLO5
                MASSLO6     MASSLO7     MASSLO8     MASSLO9     MASSLOA
                CMASLO      TMASLO      MASSGO1     MASSGO2     MASSGO3
                MASSGO4     MASSGO5     CMASGO      TMASGO      MASS  1
                MASS  2     MASS  3     MASS  4     MASS  5     MASS  6
                MASS  7     MASS  8     MASS  9     MASS  A     MASS  B
                MASS  C     MASL  1     MASL  2     MASL  3     MASL  4
                MASL  5     MASL  6     MASL  7     MASL  8     MASL  9
                MASL  A     MASG  1     MASG  2     MASG  3     MASG  4
                MASG  5
 9         1  ALLTIMNM         4         0         2
           *    ALPLK 3T       1   5     MASL  6T       0   0

----------------------RUMPF-------------------------------------------------

10         0  CYCLINIT         0       .00000000E+00
15         3  VEL   2         36         2         1
      0    0    0    0    0    0    0    0         0        12
           *  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00
           *  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00
15         2  ALPLK 3          36         0         1
      0    0    0    0    0    0    0    0        12         0
           *  7.9460E-02  7.9460E-02  7.9460E-02  0.0000E+00  7.9460E-02
              7.9460E-02  4.5390E-01  7.9460E-02  7.9460E-02  4.5390E-01
              7.9460E-02  7.9460E-02  3.1206E-01  1.0000E+00  1.0000E+00
              3.1206E-01  1.0000E+00  1.0000E+00  3.1206E-01  1.0000E+00
```

```
                1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00
                1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00
                1.0000E+00  1.0000E+00  1.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00
19          1   INTGRLVL        46          0          1
            *   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  2.7123E-18
                2.7123E-18  2.7123E-18  2.7123E-18  2.7123E-18  2.7123E-18
                5.0489E+03  2.7123E-18  2.7123E-18  7.1375E+03  5.9934E+05
                2.7123E-18  2.7123E-18  1.0167E+04  3.2396E+03  1.1876E+05
                2.7123E-18  2.7123E-18  2.7123E-18  2.7123E-18  2.7123E-18
                2.7123E-18  2.7123E-18  2.7001E+01  1.1048E-06  1.0858E+01
                2.7123E-18
20          1   TIMEALL          1          0          1
            *   0.0000E+00
21          2   ALPLK 3T         0          0          1
      1     5     0    0
            *   3.1206E-01
21          2   MASL  6T         2          0          1
      0     0     0    0
            *   2.7123E-18

10          0   CYCLPOST        38       .16110670E-02
15          3   VEL   2         36          2          1
      0     0     0    0    0    0    0    0          0         12
            *   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00 -1.2447E-04 -1.2447E-04
                0.0000E+00 -3.9459E-04 -3.9459E-04  0.0000E+00 -1.3596E-03
               -1.3596E-03 -2.1189E-02 -2.5747E-02 -4.5583E-03 -5.8814E-03
               -8.3643E-03 -2.4829E-03 -1.1595E-03 -1.7233E-03 -5.6379E-04
               -5.7036E-05 -8.3255E-05 -2.6219E-05 -2.3784E-08 -3.4102E-08
               -1.0319E-08
            *   0.0000E+00  0.0000E+00  0.0000E+00  2.1651E+00  0.0000E+00
                0.0000E+00  2.1812E+00  0.0000E+00  0.0000E+00  2.0540E-02
                0.0000E+00  0.0000E+00 -3.3894E-03 -7.0905E-04 -9.8527E-04
               -1.5932E-02 -8.2301E-04 -2.0860E-03 -6.8662E-02  3.8419E-03
               -4.1560E-04 -7.5096E-02  5.0796E-03  3.9310E-03 -1.3519E-02
                6.9608E-03  1.1045E-02  1.0116E-02  1.6071E-02  1.8583E-02
                2.6184E-02  2.7427E-02  2.8031E-02  1.7077E-02  1.7193E-02
                1.7247E-02
15          2   ALPLK 3          36          0          1
      0     0     0    0    0    0    0    0         12         0
            *   7.9460E-02  7.9460E-02  7.9460E-02  0.0000E+00  7.9460E-02
                7.9460E-02  4.5395E-01  7.9460E-02  7.9460E-02  4.5390E-01
                7.9460E-02  7.9460E-02  3.1146E-01  1.0000E+00  1.0000E+00
```

```
                3.1145E-01  1.0000E+00  1.0000E+00  3.1157E-01  1.0000E+00
                1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00
                1.0000E+00  1.0000E+00  9.9999E-01  1.0000E+00  1.0000E+00
                9.9998E-01  9.9998E-01  9.9998E-01  6.6606E-05  6.6728E-05
                6.6776E-05
19          1   INTGRLVL        46          0          1
            *   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
                0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  2.7123E-18
                2.7123E-18  2.7123E-18  2.7123E-18  2.7123E-18  3.3491E-03
                5.0489E+03  3.1060E+03  2.7123E-18  7.1376E+03  5.9623E+05
                2.7123E-18  2.7123E-18  1.0174E+04  3.2395E+03  1.1876E+05
                2.7123E-18  1.5535E-02  1.3570E-14  2.7123E-18  2.7123E-18
                2.7123E-18  2.7123E-18  1.9889E+01  2.3666E-03  1.0711E+01
                2.7123E-18
20          1   TIMEALL         38          0          1
            *   1.0000E-06  2.6800E-06  4.3600E-06  6.0400E-06  7.7200E-06
                1.0542E-05  1.3365E-05  1.6187E-05  1.9010E-05  2.3751E-05
                2.8493E-05  3.3234E-05  3.7976E-05  4.5942E-05  5.3908E-05
                6.1874E-05  6.9840E-05  8.3223E-05  9.6605E-05  1.0999E-04
                1.2337E-04  1.4585E-04  1.6834E-04  1.9082E-04  2.1330E-04
                2.5827E-04  3.0324E-04  3.4820E-04  3.9317E-04  4.8310E-04
                5.7303E-04  6.6296E-04  7.5290E-04  9.0398E-04  1.0551E-03
                1.2062E-03  1.3572E-03  1.6111E-03
21          2   ALPLK 3T         0          0          1
      1     5     0    0
            *   3.1206E-01  3.1206E-01  3.1206E-01  3.1206E-01  3.1206E-01
                3.1206E-01  3.1206E-01  3.1205E-01  3.1205E-01  3.1205E-01
                3.1205E-01  3.1205E-01  3.1205E-01  3.1204E-01  3.1204E-01
                3.1204E-01  3.1203E-01  3.1203E-01  3.1202E-01  3.1202E-01
                3.1201E-01  3.1200E-01  3.1199E-01  3.1198E-01  3.1198E-01
                3.1196E-01  3.1194E-01  3.1192E-01  3.1190E-01  3.1187E-01
                3.1183E-01  3.1180E-01  3.1177E-01  3.1171E-01  3.1165E-01
                3.1160E-01  3.1154E-01  3.1146E-01
21          2   MASL  6T         2          0          1
      0     0     0    0
            *   2.7123E-18  4.4254E-11  3.2299E-10  1.0851E-09  2.5542E-09
                6.0281E-09  1.3012E-08  2.4511E-08  4.1560E-08  7.5111E-08
                1.3317E-07  2.1825E-07  3.3533E-07  5.5336E-07  9.1257E-07
                1.4133E-06  2.0762E-06  3.2710E-06  5.1814E-06  7.7680E-06
                1.1116E-05  1.7032E-05  2.6303E-05  3.8596E-05  5.4240E-05
                8.5150E-05  1.3672E-04  2.0674E-04  2.9719E-04  4.7805E-04
                7.8447E-04  1.2096E-03  1.7723E-03  2.7911E-03  4.4496E-03
                6.7614E-03  9.8563E-03  1.5535E-02

10          0   CYCLPOST        43       .30832514E-02
15          3   VEL   2         36          2          1
```

```
      0    0    0    0    0    0    0    0          0         12
           *  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00 -1.2331E-04 -1.2331E-04
              0.0000E+00 -4.0644E-04 -4.0644E-04  0.0000E+00 -1.5791E-03
             -1.5791E-03 -3.1730E-02 -3.7800E-02 -6.0695E-03 -8.0557E-03
             -1.0930E-02 -2.8748E-03 -1.4862E-03 -2.2176E-03 -7.3138E-04
             -2.2241E-04 -3.5227E-04 -1.2986E-04 -2.2665E-07 -3.8345E-07
             -1.5680E-07
           *  0.0000E+00  0.0000E+00  0.0000E+00  3.7882E+00  0.0000E+00
              0.0000E+00  3.9280E+00  0.0000E+00  0.0000E+00  1.6374E-01
              0.0000E+00  0.0000E+00  9.7869E-03  6.4080E-03  6.1185E-03
             -2.9752E-02  1.8083E-02  1.6595E-02 -1.0205E-01  2.7590E-02
              2.1799E-02 -1.0761E-01  1.8994E-02  1.7668E-02 -2.6796E-02
              4.1243E-03  9.5828E-03 -5.8537E-03  2.1558E-03  4.9626E-03
              1.1576E-02  1.3322E-02  1.4202E-02  1.1758E-02  1.2212E-02
              1.2477E-02
15         2  ALPLK 3         36          0          1
      0    0    0    0    0    0    0    0         12          0
           *  7.9460E-02  7.9460E-02  7.9460E-02  0.0000E+00  7.9460E-02
              7.9460E-02  4.5401E-01  7.9460E-02  7.9460E-02  4.5400E-01
              7.9460E-02  7.9460E-02  3.1101E-01  1.0000E+00  1.0000E+00
              3.1097E-01  1.0000E+00  1.0000E+00  3.1133E-01  1.0000E+00
              1.0000E+00  9.9999E-01  1.0000E+00  1.0000E+00  9.9999E-01
              1.0000E+00  1.0000E+00  9.9998E-01  9.9999E-01  9.9999E-01
              9.9995E-01  9.9995E-01  9.9995E-01  1.1024E-04  1.1136E-04
              1.1201E-04
19         1  INTGRLVL        46          0          1
           *  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  2.7123E-18
              2.7123E-18  2.7123E-18  2.7123E-18  2.7123E-18  2.1379E-02
              5.0489E+03  3.1061E+03  2.7123E-18  7.1376E+03  5.9623E+05
              2.7123E-18  2.7123E-18  1.0176E+04  3.2395E+03  1.1876E+05
              2.7123E-18  1.2670E-01  1.3570E-14  2.7123E-18  2.7123E-18
              2.7123E-18  2.7123E-18  1.7854E+01  3.1005E-02  1.0610E+01
              2.7123E-18
20         1  TIMEALL          5          0          1
           *  1.8649E-03  2.1187E-03  2.3725E-03  2.7279E-03  3.0833E-03
21         2  ALPLK 3T         0          0          1
      1    5    0    0
           *  3.1137E-01  3.1129E-01  3.1121E-01  3.1111E-01  3.1101E-01
21         2  MASL  6T         2          0          1
      0    0    0    0
           *  2.4927E-02  3.8167E-02  5.5929E-02  8.4476E-02  1.2670E-01

.....................................................................
```

### A.2 Beispiel-Kompaktinformation über eine VISART-Datei

Die nachstehende Liste zeigt die von VISARTUL, LEVEL = −1, ausgedruckte Kompaktinformation über die VISART-Datei aus Beispiel A.1. (ETA steht für die Problemzyklusnummer, T steht für die Problemzeit.)

```
VISARTUL  -  VISART-RELEASE 1.22    -  OPTIONS: UNIX ON, DBLPP OFF, SCREEN
LEVEL=-1

UNFORMATTIERTE DATEI, 1-FACHE REAL-WORTLAENGE, RELEASE 1.22


CODE UND RELEASE: SIMMER-I  II 2.A    BZW.  SIMMER-III 2.A

MODUL           :                     INR308    96-11-06  10:19:32

LAUF            : inrrisc6            kleinh1   96/11/ 7  11.10.55

PROBLEMTITEL:
 :   LITTLE WORK ENERGY PROBLEM  EURO-PNC FOR S-III VER.2.A


GEBIETSDIMENSION:        2          RAUMDIMENSION   :        2
NETZ            :        1  (REGULAERES NETZ)
KO-SYSTEM       :      220  (R-Z KREISZYLINDER)
ANZAHL MASCHENMITTEN :        3 *      12
ANZAHL MASCHENHUELLEN:        4 *      13

GRUPPEN IM KOPF-PAKET (KENNZAHL, IDENTIFIKATION, ANZAHL DER WERTE):
           5  WIDTHS          36
           5  VOLUMES         36
           9  BOUNDARY         1
           9  INITIAL          1
           9  CELL  NM         2
           9  INTGRLNM        46
           9  ALLTIMNM         4

GRUPPEN IM RUMPF-PAKET (KENNZAHL, IDENTIFIKATION, ANZAHL DER WERTE):
          15  VEL   2         36
          15  ALPLK 3         36
          19  INTGRLVL        46
          20  TIMEALL
          21  ALPLK 3T
          21  MASL  6T

ANZAHL RUMPF-PAKETE:      193
VON ETA =       0, T =  0.0000E+00, BIS ETA =    1166, T =  3.0003E-01
```

## B.1 Musterprogramm zum Lesen einer VISART-Datei

Die nachstehenden Fortran-Programmzeilen sind dem Programm VISARTUL (siehe 4.3) entnommen und enthalten die Fortran-Anweisungen zum Lesen und Interpretieren einer VISART-Datei.

Die Anweisungen lesen VISART-Dateien, die auf Release 1.22 und älter beruhen.[1] Der Name der Datei wird als Argument des Programmaufrufs erwartet oder default-mäßig als „VISART“ angenommen. Die Datei darf für das Programm formatiert oder unformatiert sein, und im letzteren Fall einfache oder doppelte Genauigkeit der `REAL`-Daten haben.

Die beiden `LOGICAL`-Parameter `GRSKIP` und `DTSKIP` können dazu benutzt werden, nach dem Lesen des Kennsatzes einer Gruppe bzw. erst nach dem Lesen des Spezifikationssatzes einer Gruppe die restlichen Sätze nicht benötigter Gruppen zu überlesen.

```
      PARAMETER(NI=5,NF=10,NO=6)
      COMMON/ALL/ IZGEO,IZNOI,IZNOJ,IZNOK,IZLOC
      LOGICAL GRSKIP /.FALSE./

      CHARACTER*8 CDRELS
      CHARACTER*8 CCNAME,CCRELS,CCAUTH,CCDATE,CCTIME
      CHARACTER*8 CJNAME,CJNUMB,CJAUTH,CJDATE,CJTIME
      CHARACTER*8 CINAME,CINUMB,CIAUTH,CIDATE,CITIME
      CHARACTER*80 CIIDE1,CIIDE2
      CHARACTER*8 CZNAME,CNAME
      CHARACTER*255 FILEV

      DIMENSION L12(2)
      EQUIVALENCE (L12(1),L1),(L12(2),L2)
      REAL*8 DL1
      EQUIVALENCE (L12,DL1,RL1)
      CHARACTER*16 L1L2

      CHARACTER*8 RELEAS
      DATA RELEAS/'1.22'/

      CALL GETARG(1,FILEV)
      IF (FILEV.EQ.' ') FILEV='VISART'
      OPEN(UNIT=NF,STATUS='OLD',FILE=FILEV,FORM='UNFORMATTED')
      READ (NF,END=701) IDFORM,IDDBL,CDRELS
      IF (IDFORM.EQ.0) THEN
       IFMTP=0
       WRITE (NO,'('' UNFORMATTIERTE DATEI,'',I2,''-FACHE REAL-'',
     *            ''WORTLAENGE, RELEASE '',A8)') IDDBL,CDRELS
      ELSE
       CLOSE(NF)
```

[1] aber nicht das veraltete Format der Spezifikationssätze der Gruppen 5/15; siehe Teil 2 [1b] der VISART-Dokumentation

```
       OPEN(UNIT=NF,STATUS='OLD',FILE=FILEV,FORM='FORMATTED')
       IFMTP=1
       READ (NF,800,END=701) IDFORM,IDDBL,CDRELS
       WRITE (NO,'('' FORMATTIERTE DATEI, RELEASE '',A8)') CDRELS
      ENDIF

C=====KOPF-PAKET ====================================================

C-----OBLIGATORISCHE KOPF-GRUPPEN -----------------------------------

      IF (IFMTP.EQ.0) THEN
       READ (NF,END=701) N1,M1,CCNAME,CCRELS,CCAUTH,CCDATE,CCTIME
      ELSE
       READ (NF,801,END=701) N1,M1,CCNAME,CCRELS,CCAUTH,CCDATE,CCTIME
      ENDIF
      IF (N1.NE.1) GO TO 706

      IF (IFMTP.EQ.0) THEN
       READ (NF,END=701) N2,M2,CJNAME,CJNUMB,CJAUTH,CJDATE,CJTIME
      ELSE
       READ (NF,801,END=701) N2,M2,CJNAME,CJNUMB,CJAUTH,CJDATE,CJTIME
      ENDIF
      IF (N2.NE.2) GO TO 706

      IF (IFMTP.EQ.0) THEN
       READ (NF,END=701) N3,M3,CINAME,CINUMB,CIAUTH,CIDATE,CITIME
      ELSE
       READ (NF,801,END=701) N3,M3,CINAME,CINUMB,CIAUTH,CIDATE,CITIME
      ENDIF
      IF (N3.NE.3) GO TO 706
      IF (IFMTP.EQ.0) THEN
       READ (NF,END=701) (CIIDE1(I:I+7),I=1,73,8)
       READ (NF,END=701) (CIIDE2(I:I+7),I=1,73,8)
      ELSE
       READ (NF,802,END=701) (CIIDE1(I:I+7),I=1,73,8)
       READ (NF,802,END=701) (CIIDE2(I:I+7),I=1,73,8)
      ENDIF

      IF (IFMTP.EQ.0) THEN
       READ (NF,END=701) N,M,CZNAME,IZDIM,IZGEO,IZSYS
      ELSE
       READ (NF,810,END=701) N,M,CZNAME,IZDIM,IZGEO,IZSYS
      ENDIF
      IF (N.NE.4) GO TO 706
      IZMSH=MOD(IZDIM-1,3)+1
      NPREVK=0
C- - -SPEZIFIKATOREN UND DATEN - - - - - - - - - - - - - - - - - - - - -
      CALL GRUPPE(IFMTP,IDDBL,4,M,0,IZMSH,1)
```

```
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

C-----FAKULTATIVE KOPF-GRUPPEN ----------------------------------------

   50 IF (IFMTP.EQ.0) THEN
       READ (NF,END=700) N,M,CNAME,L,L1,L2
      ELSE
       READ (NF,811,END=700) N,M,CNAME,L,L1L2
      ENDIF
      IF (N.EQ.10) GO TO 100
      IF (IFMTP.NE.0) READ (L1L2,'(2I8)') L1,L2
      IF (N.EQ.7.AND..NOT.NPREVK.EQ.6)
     * WRITE (NO,990) NPREVK
      IF (N.EQ.6) NPREVK=N
      IF (N.NE.7.AND.L.LE.0) GO TO 707
      IF (.NOT.GRSKIP) THEN
C- - - SPEZIFIKATOREN UND DATEN - - - - - - - - - - - - - - - - - - - -
       IF (N.EQ.5) THEN
        CALL GRUPPE(IFMTP,IDDBL,5,M,L,L1,L2)
       ELSEIF (N.EQ.6) THEN
        CALL GRUPPE(IFMTP,IDDBL,6,M,L,IZMSH,L2)
        LL=L
       ELSEIF (N.EQ.7) THEN
        CALL GRUPPE(IFMTP,IDDBL,7,M,LL,L1,L2)
       ELSEIF (N.EQ.9) THEN
        CALL GRUPPE(IFMTP,IDDBL,9,M,L,L1,L2)
       ELSE
        WRITE (NO,994)
  994   FORMAT('0FEHLER: UNBEKANNTE KOPF-GRUPPE')
        STOP 99
       ENDIF
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      ELSE
       IF (M.LE.0) GO TO 50
       DO 51 I=1,M
       IF (IFMTP.EQ.0) THEN
        READ (NF,END=703)
       ELSE
        READ (NF,800,END=703)
       ENDIF
   51  CONTINUE
      ENDIF
      GO TO 50

C=====RUMPF-PAKETE ====================================================

C-----OBLIGATORISCHE ZYKLUS-GRUPPE -------------------------------------
```

```
  100 CONTINUE
      IYCC=L
      IF (IFMTP.EQ.0) THEN
       IF (IDDBL.EQ.1) THEN
        YTIME=RL1
       ELSE
        YTIME=DL1
       ENDIF
      ELSE
       READ (L1L2,'(E16.8)') YTIME
      ENDIF
      NPREVR=0

C-----FAKULTATIVE RUMPF-GRUPPEN ---------------------------------------

  150 IF (IFMTP.EQ.0) THEN
       READ (NF,END=700) N,M,CNAME,L,L1,L2
      ELSE
       READ (NF,811,END=700) N,M,CNAME,L,L1L2
      ENDIF
      IF (N.EQ.10) GO TO 100
      IF (IFMTP.NE.0) READ (L1L2,'(2I8)') L1,L2
      IF (N.EQ.17.AND..NOT.NPREVR.EQ.16.OR.
     *    N.EQ.21.AND..NOT.NPREVR.EQ.20)
     * WRITE (NO,990) NPREVR
      IF (N.EQ.16.OR.N.EQ.20) NPREVR=N
      IF (N.NE.17.AND.N.NE.21.AND.L.LE.0) GO TO 707
      IF (.NOT.GRSKIP) THEN
C- - - SPEZIFIKATOREN UND DATEN - - - - - - - - - - - - - - - - - - -
       IF (N.EQ.15) THEN
        CALL GRUPPE(IFMTP,IDDBL,15,M,L,L1,L2)
       ELSEIF (N.EQ.16) THEN
        CALL GRUPPE(IFMTP,IDDBL,16,M,L,IZMSH,L2)
        LL=L
       ELSEIF (N.EQ.17) THEN
        CALL GRUPPE(IFMTP,IDDBL,17,M,LL,L1,L2)
       ELSEIF (N.EQ.19) THEN
        CALL GRUPPE(IFMTP,IDDBL,19,M,L,L1,L2)
       ELSEIF (N.EQ.20) THEN
        CALL GRUPPE(IFMTP,IDDBL,20,M,L,1,L2)
        LL=L
       ELSEIF (N.EQ.21) THEN
        LL0=LL
        IF (L.EQ.1.OR.L.EQ.3) LL0=-LL
        CALL GRUPPE(IFMTP,IDDBL,21,M,LL0,L1,L2)
       ELSE
        WRITE (NO,995)
  995   FORMAT('0FEHLER: UNBEKANNTE RUMPF-GRUPPE')
```

```
        STOP 99
       ENDIF
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      ELSE
       IF (M.LE.0) GO TO 150
       DO 151 I=1,M
       IF (IFMTP.EQ.0) THEN
        READ (NF,END=703)
       ELSE
        READ (NF,800,END=703)
       ENDIF
  151  CONTINUE
      ENDIF
      GO TO 150

C=====DATEI-ENDE ========================================================

  700 STOP

  701 WRITE (NO,'(''0FEHLER: DATEI-ENDE IN OBLIGATORISCHEN KOPF-'',
     *            ''GRUPPEN'')')
      STOP 99
  703 WRITE (NO,'(''0FEHLER: DATEI-ENDE INNERHALB EINER GRUPPE'')')
      STOP 99
  706 WRITE (NO,'(''0FEHLER: FEHLPLAZIERTE KOPF-GRUPPE'')')
      STOP 99
  707 WRITE (NO,'(''0FEHLER: LEERE GRUPPE'')')
      STOP 99

  800 FORMAT(2I8,A8)
C.....BESCHREIBUNG:
  801 FORMAT(2I8,5A8)
  802 FORMAT(10A8)
C.....ZYKLUS-KENNSATZ:
  811 FORMAT(2I8,A8,I8,A16)
C.....SONSTIGER KENNSATZ:
  810 FORMAT(2I8,A8,3I8)

  990 FORMAT('0WARNUNG: GRUPPE INKOMPATIBEL MIT VORHERIGER GRUPPE',I3)

      END


      SUBROUTINE GRUPPE(IFMTP,IDDBL,N,M,L,L1,L2)
C-----LIEST UND PRUEFT SPEZIFIKATIONS- UND DATENSAETZE.

      PARAMETER (NF=10,NO=6,LMAX=120000)
      COMMON/ALL/ IZGEO,IZNOI,IZNOJ,IZNOK,IZLOC
```

```
      LOGICAL DTSKIP /.FALSE./

      REAL*8 DANGI,DANGJ,DANGK,DAI,DAJ,DAK

      CHARACTER*8 CKOM
      DIMENSION CKOM(LMAX,3)
      DIMENSION AKOM(LMAX,3)
      REAL*8 DAKOM
      DIMENSION DAKOM(LMAX,3)
      DIMENSION IKOM(LMAX,3)
      LOGICAL LKOM
      DIMENSION LKOM(LMAX,3)
      EQUIVALENCE (CKOM,AKOM,DAKOM,ICOM,LCOM)

C-----SPEZIFIKATIONSSAETZE LESEN ---------------------------------------

      L0=SIGN(1,L)
      L=ABS(L)
      LI=L
      LJ=L
      LK=L
      J=M
      IF (IFMTP.EQ.0) THEN
       IF (N.EQ.4) THEN
        IF (IDDBL.EQ.1) THEN
         READ (NF,END=703) LI,LJ,LK,ILOC,ANGI,ANGJ,ANGK
        ELSE
         READ (NF,END=703) LI,LJ,LK,ILOC,DANGI,DANGJ,DANGK
         ANGI=DANGI
         ANGJ=DANGJ
         ANGK=DANGK
        ENDIF
       ELSEIF (N.EQ.5.OR.N.EQ.15) THEN
        READ (NF,END=703) IDIM,IPRT,IKORI1,IKORI2,IKORJ1,IKORJ2,
     *                    IKORK1,IKORK2,IORD,ILOC
       ELSEIF (N.EQ.21) THEN
        IF (L0.GE.0) THEN
         READ (NF,END=703) IA,JA,KA,ILOC
        ELSE
         IF (IDDBL.EQ.1) THEN
          READ (NF,END=703) AI,AJ,AK,ILOC
         ELSE
          READ (NF,END=703) DAI,DAJ,DAK,ILOC
          AI=DAI
          AJ=DAJ
          AK=DAK
         ENDIF
        ENDIF
```

```
       ENDIF
      ELSE
       IF (N.EQ.4) THEN
        READ (NF,803,END=703) LI,LJ,LK,ILOC,ANGI,ANGJ,ANGK
       ELSEIF (N.EQ.5.OR.N.EQ.15) THEN
        READ (NF,806,END=703) IDIM,IPRT,IKORI1,IKORI2,IKORJ1,IKORJ2,
     *                        IKORK1,IKORK2,IORD,ILOC
       ELSEIF (N.EQ.21) THEN
        IF (L0.GE.0) THEN
         READ (NF,804,END=703) IA,JA,KA,ILOC
        ELSE
         READ (NF,805,END=703) AI,AJ,AK,ILOC
        ENDIF
       ENDIF
      ENDIF
      IF (N.EQ.4) THEN
       IZNOI=LI
       IZNOJ=LJ
       IZNOK=LK
       IZLOC=ILOC
       J=J-1
      ELSEIF (N.EQ.5.OR.N.EQ.15) THEN
       J=J-1
      ELSEIF (N.EQ.21) THEN
       J=J-1
      ENDIF

      IF ((N.EQ.4.AND.IZGEO.NE.1).OR.(N.NE.4.AND.L2.LT.0)) RETURN

C-----DATENSAETZE LESEN ------------------------------------------------

      IF (DTSKIP) THEN

       IF (J.GT.0) THEN
        DO 5 I=1,J
        IF (IFMTP.EQ.0) THEN
         READ (NF,END=703)
        ELSE
         READ (NF,803,END=703)
        ENDIF
    5   CONTINUE
       ENDIF

       RETURN

      ELSE

       IF (L.GT.LMAX) GO TO 708
```

```
      IF (L2.EQ.0) THEN
       IF (IFMTP.EQ.0) THEN
        READ (NF,END=703) (IKOM(I,1),I=1,LI)
       ELSE
        READ (NF,820,END=703) (IKOM(I,1),I=1,LI)
       ENDIF
      ELSEIF (L2.EQ.1) THEN
       IF (IFMTP.EQ.0) THEN
        IF (IDDBL.EQ.1) THEN
         READ (NF,END=703) (AKOM(I,1),I=1,LI)
        ELSE
         READ (NF,END=703) (DAKOM(I,1),I=1,LI)
         DO 6 I=1,LI
         AKOM(I,1)=DAKOM(I,1)
6        CONTINUE
        ENDIF
       ELSE
        READ (NF,821,END=703) (AKOM(I,1),I=1,LI)
       ENDIF
      ELSEIF (L2.EQ.2) THEN
       IF (IFMTP.EQ.0) THEN
        READ (NF,END=703) (CKOM(I,1),I=1,LI)
       ELSE
        READ (NF,822,END=703) (CKOM(I,1),I=1,LI)
       ENDIF
      ELSEIF (L2.EQ.3) THEN
       IF (IFMTP.EQ.0) THEN
        READ (NF,END=703) (LKOM(I,1),I=1,LI)
       ELSE
        READ (NF,823,END=703) (LKOM(I,1),I=1,LI)
       ENDIF
      ELSE
       GO TO 707
      ENDIF

      IF (L1.GT.1) THEN
       IF (L2.EQ.0) THEN
        IF (IFMTP.EQ.0) THEN
         READ (NF,END=703) (IKOM(I,2),I=1,LJ)
        ELSE
         READ (NF,820,END=703) (IKOM(I,2),I=1,LJ)
        ENDIF
       ELSEIF (L2.EQ.1) THEN
        IF (IFMTP.EQ.0) THEN
         IF (IDDBL.EQ.1) THEN
          READ (NF,END=703) (AKOM(I,2),I=1,LJ)
         ELSE
```

```
        READ (NF,END=703) (DAKOM(I,2),I=1,LJ)
        DO 7 I=1,LJ
        AKOM(I,2)=DAKOM(I,2)
7       CONTINUE
       ENDIF
      ELSE
       READ (NF,821,END=703) (AKOM(I,2),I=1,LJ)
      ENDIF
     ELSEIF (L2.EQ.2) THEN
      IF (IFMTP.EQ.0) THEN
       READ (NF,END=703) (CKOM(I,2),I=1,LJ)
      ELSE
       READ (NF,822,END=703) (CKOM(I,2),I=1,LJ)
      ENDIF
     ELSEIF (L2.EQ.3) THEN
      IF (IFMTP.EQ.0) THEN
       READ (NF,END=703) (LKOM(I,2),I=1,LJ)
      ELSE
       READ (NF,823,END=703) (LKOM(I,2),I=1,LJ)
      ENDIF
     ELSE
      GO TO 707
     ENDIF
    ENDIF

    IF (L1.GT.2) THEN
     IF (L2.EQ.0) THEN
      IF (IFMTP.EQ.0) THEN
       READ (NF,END=703) (IKOM(I,3),I=1,LK)
      ELSE
       READ (NF,820,END=703) (IKOM(I,3),I=1,LK)
      ENDIF
     ELSEIF (L2.EQ.1) THEN
      IF (IFMTP.EQ.0) THEN
       IF (IDDBL.EQ.1) THEN
        READ (NF,END=703) (AKOM(I,3),I=1,LK)
       ELSE
        READ (NF,END=703) (DAKOM(I,3),I=1,LK)
        DO 8 I=1,LK
        AKOM(I,3)=DAKOM(I,3)
8       CONTINUE
       ENDIF
      ELSE
       READ (NF,821,END=703) (AKOM(I,3),I=1,LK)
      ENDIF
     ELSEIF (L2.EQ.2) THEN
      IF (IFMTP.EQ.0) THEN
       READ (NF,END=703) (CKOM(I,3),I=1,LK)
```

```
         ELSE
          READ (NF,822,END=703) (CKOM(I,3),I=1,LK)
         ENDIF
        ELSEIF (L2.EQ.3) THEN
         IF (IFMTP.EQ.0) THEN
          READ (NF,END=703) (LKOM(I,3),I=1,LK)
         ELSE
          READ (NF,823,END=703) (LKOM(I,3),I=1,LK)
         ENDIF
        ELSE
         GO TO 707
        ENDIF
       ENDIF

      ENDIF

      RETURN

  703 WRITE (NO,'(''0FEHLER: DATEI-ENDE INNERHALB EINER GRUPPE'')')
      STOP 99
  707 WRITE (NO,'(''0FEHLER: UNBEKANNTER DATENTYP'')')
      STOP 99
  708 WRITE (NO,'(''0FEHLER: FELD-DIMENSION'',I8,'' UEBERSCHRITTEN'')')
     *      LMAX
      STOP 99

C.....SPEZIFIKATIONEN:
  803 FORMAT(4I8,3E16.8)
  804 FORMAT(6I8)
  805 FORMAT(3E16.8,I8)
  806 FORMAT(10I8)
C.....DATEN:
  820 FORMAT(10I8)
  821 FORMAT(5E16.8)
  822 FORMAT(10A8)
  823 FORMAT(10L8)

      END
```

### B.2 Muster des Steuerungsteils eines Computer-Codes mit VISART-Ausgabe

Die nachstehende Liste zeigt, wie der Steuerungsteil eines Computer-Codes aufgebaut sein könnte, der Postprocessor-Ausgabe im VISART-Format erzeugt.

Die Routinen `VIHEAD` und `VIBODY` (siehe B.3) werden zur Ausgabe des Kopf-Pakets bzw. der Rumpf-Pakete aufgerufen.

Die Routinen `VIREAD` und `VICYCL` (siehe B.3) werden nur aufgerufen, wenn auch Zeitfunktionsgrößen (Gruppen 20 und 21) geschrieben werden sollen.

```
      PROGRAM XXXX
*CALL IMPDBL   (siehe Liste B.3)
*CALL PARAMETR (siehe Liste B.3)
*CALL ALL      (siehe Liste B.3)
      CHARACTER*8 EXMACH,EXUSER,EXDATE,EXTIME

C-----Angaben ueber den Prozess speichern und ausdrucken --------------------

C     Abfragen von Workstation- und Benutzername, Datum und Uhrzeit:
      CALL MACNAM(EXMACH)
      CALL JOBNAM(EXUSER)
      CALL DATE(EXDATE)
      CALL TIME(EXTIME)

C     Speichern bzw. Einlesen der Angaben ueber den Erstlauf:
      ................
      IF (Erstlauf) THEN
       CMACN(1)=EXMACH
       CJOBN(1)=EXUSER
       CADAT(1)=EXDATE
       CLTIM(1)=EXTIME
      ELSE
       ..... (Commons aus der Restart-Datei lesen)
      ENDIF
C     Speichern der Angaben ueber den gegenwaertigen Lauf:
      CMACN(2)=EXMACH
      CJOBN(2)=EXUSER
      CADAT(2)=EXDATE
      CLTIM(2)=EXTIME
C     Titelseite drucken, Problemeingabe wiedergeben, usw.:
      ................

C-----Problemeingabe verarbeiten; Rechengang initialisieren -----------------

      IF (Restart-Lauf) THEN
       ..... (Restart-Bezeichnung drucken)
       CALL VIHEAD
      ELSE
       ..... (Problemeingabe lesen und verarbeiten; initialisieren)
```

```
       CALL VIREAD
       CALL VIHEAD
       NCYC=0
       TIME=0.
       CALL VICYCL(1)
       CALL VIBODY(1,'CYCLINIT')
      ENDIF
      IVIACC=0

C-----Schleife ueber die Problemzeitschritte des Rechengangs ----------------

    5 CONTINUE
      ..... (Rechnung im Intervall t,...,t+dt)
      NCYC=NCYC+1
      TIME=TIME+DTIME
      IVIACC=IVIACC+1
      CALL VICYCL(IVIACC)
      IF (Problem durchgerechnet) THEN
       GO TO 7
      ENDIF
      IF (Lauf abgebrochen) THEN
       GO TO 8
      ENDIF
      IF (Postprocessor-Ausgabe faellig) THEN
       CALL VIBODY(IVIACC,'CYCLPOST')
       IVIACC=0
      ENDIF
      GO TO 5

C-----Vor dem Stop die Postprocessor- und Restart-Ausgabe schreiben ---------

    7 CALL VIBODY(IVIACC,'CYCLFINI')
      ..... (Restart-Ausgabe)
      WRITE (NDIAGO,'('0JOB FINISHED AT PRESCRIBED PROBLEM TIME')')
      GO TO 9
    8 CALL VIBODY(IVIACC,'CYCLTERM')
      ..... (Restart-Ausgabe)
      WRITE (NDIAGO,'('0JOB TERMINATED BECAUSE CPU TIME EXHAUSTED')')
    9 STOP
      END
```

### B.3 Musterprogramm(teile) zum Erzeugen einer VISART-Datei

Die nachstehende Liste zeigt, wie die Programmteile zum Erzeugen einer Postprocessor-Datei im VISART-Format für ein zweidimensionales reguläres Netz aufgebaut sein könnten. Die VISART-Datei wird hier unformatiert, mit einfacher Genauigkeit der `REAL`-Daten, geschrieben, während der Code selbst mit doppelter Genauigkeit rechnen soll. Deswegen werden die vom Code berechneten `REAL*8`-Größen zunächst in das `REAL*4`-Feld `FV` umgespeichert und erst von dort auf die Datei geschrieben.

Die den Routinen gemeinsamen Parameter, Commons usw. sind der Übersichtlichkeit halber in HISTORIAN-Manier den Routinen vorangestellt worden; sie sind in einem Preprocessing-Schritt in die Routinen einzufügen. Die Parameter `RELEAS` und `VERSIO` sollen beim Preprocessing mit dem Code-Namen und -Release bzw. mit Autorenname, Datum und Uhrzeit der Erstellung des ausführbaren Binärprogramms gefüllt werden.

Die Routine `VIDATA` initialisiert die Common-Größen. Die Routinen `VIHEAD` und `VIBODY` dienen zur Ausgabe des Kopf-Pakets bzw. der Rumpf-Pakete.

Die in `VIBODY` gerufenen Routinen `VISAV` und `VISAS` schreiben die Gruppen 15 für vektorielle bzw. skalare Größen.

Die Routinen `VIREAD` und `VICYCL` werden nur benötigt, wenn auch Zeitfunktionsgrößen (Gruppen 20 und 21) geschrieben werden sollen. `VIREAD` liest die Steuereingabe zur Auswahl der zu schreibenden Zeitfunktionsgrößen (Subgruppen 21). `VICYCL` sammelt diese Größen für jeden Zyklus und schreibt sie im nächsten Rumpf-Paket aus.

Als Größen über dem Netz oder in einzelnen Maschen desselben sind für ein Strömungsfeld die Dichte `RHO`, der Druck `P` und die Geschwindigkeitskomponenten `U` und `V` (alle in den Maschenmitten), sowie die Geschwindigkeitskomponenten `US` und `VS` auf den Maschenhüllen in den beiden Koordinatenrichtung, vorgesehen. Aus den Geschwindigkeitskomponenten in den Maschenmitten wird der Geschwindigkeitsvektor `VEL` zusammengesetzt.

```
*COMDECK IMPDBL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

*COMDECK PARAMETR
      CHARACTER*16 RELEAS
      CHARACTER*24 VERSIO
      PARAMETER (RELEAS='xxxx    n.n')
      PARAMETER (VERSIO='UUUUUU  YYMMDD  HHMMSS')
      PARAMETER (NINPUT=5,NDIAGO=6,NPOSTO=10,...)
      PARAMETER (IBMAX=...,JBMAX=...,FVMAX=...)
      PARAMETER (IJBMAX=IBMAX*JBMAX)
      PARAMETER (IPJBMX=(IBMAX+1)*JBMAX)
      PARAMETER (IJPBMX=IBMAX*(JBMAX+1))
      PARAMETER (NSD=4,NVD=1,MAXPQU=...,MAXTQU=...,MAXACC=...)

*COMDECK CONST
      COMMON /CCONST/ SIDENT(NSD),VIDENT(NVD),...
      CHARACTER*8  SIDENT,VIDENT,...

*COMDECK ALL
      COMMON /IALL/ IB,JB,NCYC
     *             ,IPTQUA,ITQUA(MAXTQU),JTQUA(MAXTQU),...
      COMMON /RALL/ RHO(IJBMAX),P(IJBMAX),US(IPJBMX),VS(IJPBMX)
     *             ,U(IJBMAX),V(IJBMAX)
     *             ,TIME,...
      COMMON /CALL/ CMACN(2),CJOBN(2),CADAT(2),CLTIM(2)
     *             ,TQUANT(MAXTQU),...
      CHARACTER*8  CMACN,CJOBN,CADAT,CLTIM,TQUANT,...

*COMDECK WORK
      COMMON /WORK/ VISACC(MAXACC),FV(FVMAX)
      REAL*4 VISACC,FV


*DECK VISART
      BLOCK DATA VIDATA
*CALL IMPDBL
*CALL PARAMETR
*CALL CONST
*CALL ALL
      DATA TQUANT/MAXTQU*' '/
      DATA SIDENT/'RHO    ','P       ','US      ','VS      '/
      DATA VIDENT/'VEL    '/
      END


      SUBROUTINE VIREAD
C*****Verarbeitung der Steuereingabe fuer Zeitfunktionsgroessen ********
```

```
*CALL IMPDBL
*CALL PARAMETR
*CALL ALL
      ..............................................................
C-----Hier Eingabe der Strings usw. fuer die Zeitfunktionsgroessen
C               IPTQUA
C               TQUANT(1)        ITQUA(1)        JTQUA(1)
C               ......           .....           .....
C               TQUANT(IPTQUA)   ITQUA(IPTQUA)   JTQUA(IPTQUA)
      ..............................................................
      RETURN
      END


      SUBROUTINE VIHEAD
C*****Ausgabe des Kopf-Pakets der VISART-Datei ************************
*CALL IMPDBL
*CALL PARAMETR
*CALL ALL
*CALL WORK
      DATA NULL/0/
      CHARACTER*8 STANDR/'1.22'/,CAUTH/' '/,CDATE/' '/,CTIME/' '/
      CHARACTER*8 CMACH/' '/,BLANK/' '/,CNAME

      IDFORM=0
      IDDBL=1
      WRITE (NPOSTO) IDFORM,IDDBL,STANDR

C-----Schreiben der Beschreibungs-Gruppen -----------------------------

      KK=1
      MM=0
      CAUTH=VERSIO(1:8)
      CDATE=VERSIO( 9:10)//'-'//VERSIO(11:12)//'-'//VERSIO(13:14)
      CTIME=VERSIO(17:18)//':'//VERSIO(19:20)//':'//VERSIO(21:22)
      WRITE (NPOSTO) KK,MM,RELEAS,CAUTH,CDATE,CTIME

      KK=2
      MM=0
      CMACH=CMACN(2)
      CAUTH=CJOBN(2)
      CDATE=CADAT(2)
      CTIME=CLTIM(2)
      WRITE (NPOSTO) KK,MM,CMACH,BLANK,CAUTH,CDATE,CTIME

      KK=3
      MM=2
      IF (CADAT(1).EQ.CADAT(2).AND.CLTIM(1).EQ.CLTIM(2)) THEN
```

```
       CMACH=BLANK
       CAUTH=BLANK
       CDATE=BLANK
       CTIME=BLANK
      ELSE
       CMACH=CMACN(1)
       CAUTH=CJOBN(1)
       CDATE=CADAT(1)
       CTIME=CLTIM(1)
      ENDIF
      WRITE (NPOSTO) KK,MM,CMACH,BLANK,CAUTH,CDATE,CTIME
      WRITE (NPOSTO) (BLANK,I=1,10)
      WRITE (NPOSTO) (BLANK,I=1,10)

C-----Schreiben der Geometrie-Gruppe ----------------------------------
      KK=4
      MM=3
      CNAME='GEOMETRY'
      IZDIM=2
      IZGEO=1
      IZSYS=210
      IZLOC=33
      WRITE (NPOSTO) KK,MM,CNAME,IZDIM,IZGEO,IZSYS
      IBP1=IB+1
      JBP1=JB+1
      FV(1)=0.
      WRITE (NPOSTO) IBP1,JBP1,NULL,IZLOC,FV(1),FV(1),FV(1)
      DO 10 I=0,IB
      FV(I+1)=X(I)
   10 CONTINUE
      WRITE (NPOSTO) (FV(I),I=1,IBP1)
      DO 11 J=0,JB
      FV(J+1)=Y(J)
   11 CONTINUE
      WRITE (NPOSTO) (FV(J),J=1,JBP1)

      RETURN
      END


      SUBROUTINE VIBODY(KCYC,CYNAME)
C*****Ausgabe eines Rumpf-Pakets der VISART-Datei **********************
*CALL IMPDBL
*CALL PARAMETR
*CALL CONST
*CALL ALL
      CHARACTER*8 CYNAME
      REAL*4 F0
```

```
C-----Schreiben der Zyklus-Gruppe ------------------------------------
      KK=10
      MM=0
      F0=TIME
      WRITE (NPOSTO) KK,MM,CYNAME,NCYC,F0,0

C-----Schreiben der Skalargroessen als Gruppen 15 ---------------------
      CALL VISAS(SIDENT(1),RHO,0)
      CALL VISAS(SIDENT(2),P,0)
      CALL VISAS(SIDENT(3),US,1)
      CALL VISAS(SIDENT(4),VS,2)
C-----Schreiben einer Vektorgroesse als Gruppe 15 ---------------------
      CALL VISAV(VIDENT(1),U,V)

      CALL VICYCL(-KCYC)
      RETURN
      END


      SUBROUTINE VISAS(LDCON,P,L)
C*****Schreiben einer Skalargroesse als Gruppe 15 **********************
*CALL IMPDBL
*CALL PARAMETR
*CALL ALL
*CALL WORK
      DIMENSION P(I*)
      CHARACTER*8 LDCON

      KK=15
      MM=2
      ISKOM=0
      ISREP=1
      ISORD=12
      IF (L.EQ.0) THEN
       ISLOC=0
       IJM=IB*JB
      ELSEIF (L.EQ.1) THEN
       ISLOC=11
       IJM=(IB+1)*JB
      ELSEIF (L.EQ.2) THEN
       ISLOC=22
       IJM=IB*(JB+1)
      ENDIF
      WRITE (NPOSTO) KK,MM,LDCON,IJM,ISKOM,ISREP
      WRITE (NPOSTO) 0,0,0,0,0,0,0,0,ISORD,ISLOC
      DO 1 IJ=1,IJM
      FV(IJ)=P(IJ)
```

```
    1 CONTINUE
      WRITE (NPOSTO) (FV(IJ),IJ=1,IJM)
      RETURN
      END


      SUBROUTINE VISAV(LDCON,U,V)
C*****Schreiben einer Vektorgroesse als Gruppe 15 **********************
*CALL IMPDBL
*CALL PARAMETR
*CALL ALL
*CALL WORK
      DIMENSION U(*),V(*)
      CHARACTER*8 LDCON

      KK=15
      MM=3
      ISKOM=2
      ISREP=1
      ISORD=12
      ISLOC=0
      IJM=IB*JB
      WRITE (NPOSTO) KK,MM,LDCON,IJM,ISKOM,ISREP
      WRITE (NPOSTO) 0,0,0,0,0,0,0,0,ISORD,ISLOC
      DO 1 IJ=1,IJM
      FV(IJ)=U(IJ)
    1 CONTINUE
      WRITE (NPOSTO) (FV(IJ),IJ=1,IJM)
      DO 2 IJ=1,IJM
      FV(IJ)=V(IJ)
    2 CONTINUE
      WRITE (NPOSTO) (FV(IJ),IJ=1,IJM)
      RETURN
      END


      SUBROUTINE VICYCL(KCYC)
C*****Ansammeln bzw. Schreiben der gewuenschten Zeitfunktionsgroessen **
C*****als Gruppen 20/21 ************************************************
*CALL IMPDBL
*CALL PARAMETR
*CALL CONST
*CALL ALL
*CALL WORK
      REAL*4 FI,FJ,FDUM
      CHARACTER*8 TDCON/'TIMEALL'/,LDCON

      IF (IPTQUA.EQ.0) RETURN
```

```
      IF (KCYC.GT.0) THEN

C-----Ansammeln der Groessen in Feld VISACC ----------------------------
       IF (KCYC*(IPTQUA+1).GT.MAXACC) THEN
        WRITE(NDIAGO,307)
  307   FORMAT('0VICYCL: ERROR - TOO MANY CYCLEWISE PLOT QUANTITIES'/' ')
        STOP
       ENDIF
       VISACC((KCYC-1)*(IPTQUA+1)+1)=T
       DO 400 M=1,MAXTQU
       IF (TQUANT(M).EQ.' ') GO TO 490
       I=ITQUA(M)
       J=JTQUA(M)
       DO 420 II=1,NSD
       IF (TQUANT(M).EQ.SIDENT(II)) THEN
        GO TO (501,502,503,504) II
  501   VISACC((KCYC-1)*(IPTQUA+1)+M+1)=RHO((J-1)*IB+I)
        GO TO 400
  502   VISACC((KCYC-1)*(IPTQUA+1)+M+1)=P((J-1)*IB+I)
        GO TO 400
  503   VISACC((KCYC-1)*(IPTQUA+1)+M+1)=US((J-1)*(IB+1)+I+1)
        GO TO 400
  504   VISACC((KCYC-1)*(IPTQUA+1)+M+1)=VS(J*IB+I)
        GO TO 400
       ENDIF
  420  CONTINUE
       WRITE (NDIAGO,480) TQUANT(M)
  480  FORMAT('0VICYCL: POSTPROCESSOR QUANTITY ',A8,' NOT AVAILABLE;',
     *        ' SKIPPED'/' ')
  400  CONTINUE
  490  CONTINUE
      ELSE

C-----Schreiben der Groessen aus Feld VISACC ---------------------------
       LPL=ABS(KCYC)
       KK=20
       MM=1
       IFREP=1
       WRITE (NPOSTO) KK,MM,TDCON,LPL,0,IFREP
       WRITE (NPOSTO) (VISACC(II),II=1,LPL*(IPTQUA+1),(IPTQUA+1))
       DO 800 M=1,MAXTQU
       IF (TQUANT(M).EQ.' ') GO TO 890
       LDCON=TQUANT(M)
       LDCON(8:8)='T'
       DO 820 II=1,NSD
       IF (TQUANT(M).EQ.SIDENT(II)) THEN
        IFTYP=1
        IF (II.LE.2) THEN
```

```
       IFLOC=0
      ELSEIF (II.EQ.3) THEN
       IFLOC=11
      ELSEIF (II.EQ.4) THEN
       IFLOC=22
      ENDIF
      FI=ITQUA(M)
      FJ=JTQUA(M)
      GO TO 840
     ENDIF
820  CONTINUE
     WRITE (NDIAGO,880) TQUANT(M)
880  FORMAT('0',128('*')/
   *        ' VICYCL: POSTPROCESSOR QUANTITY ',A8,' NOT AVAILABLE;',
   *        ' SKIPPED'/
   *        ' ',128('*')/' ')
     GO TO 800
840  FDUM=0.
     KK=21
     MM=2
     IFKOM=0
     IFREP=1
     WRITE (NPOSTO) KK,MM,LDCON,IFTYP,IFKOM,IFREP
     WRITE (NPOSTO) FI,FJ,FDUM,IFLOC
     WRITE (NPOSTO) (VISACC(II),II=M+1,LPL*(IPTQUA+1),(IPTQUA+1))
800  CONTINUE
890  CONTINUE

    ENDIF
    RETURN
    END
```

## Danksagung

Es ist mir ein Bedürfnis, den folgenden Personen zu danken:

- dem Institutsleiter des INR, Prof. Dr. G. Keßler, und den Abteilungsleitern Dr. R. Fröhlich, Dr. E.A. Fischer, Dr. H. Jacobs, für den Freiraum und die Zeit, die mir für die Entwicklung des Konzepts zugestanden wurde;
- Frau S. Hirmer, die an der Entwicklung der ersten Fassung des Programms VISAPTER mit der Anpassung an das Visualisierungssystem AVS (Field Data) beteiligt war;
- den Kollegen aus dem INR und IRS, die durch intensive Anwendung und ständig neue Wünsche die Entwicklung des Programms VISAPTER vorantrieben; ganz besonders auch Herrn M. Linder, MATA zur Ausbildung im IRS, der mehrere konstruktive Vorschläge einbrachte und der das Vergnügen hatte, die mit heißer Nadel gestrickten Änderungen und Erweiterungen auszutesten;
- den Kollegen Dr. C. Broeders und E. Stein, INR, die die Hard- und Software im Institut am Laufen und auf dem neusten Stand halten, und die als letzte Zuflucht bei Problemen mit UNIX, Tcl/Tk, Tecplot usw. noch nie versagt haben;
- allen anderen Kollegen, die das VISART-Konzept bei der Auswertung ihrer Code-Ergebnisse benutzen und damit dieser Arbeit ihren Sinn geben;
- und schließlich Herrn Götzmann, INR, der die Zeichnungen zu diesem Bericht anfertigte.

S. Kleinheins