Forschungszentrum Karlsruhe
Technik und Umwelt

Wissenschaftliche Berichte
FZKA 5997

Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen

Teil 3: VISAPTER – ein Programm zum Anschluß einiger Visualisierungssysteme an VISART-Dateien

S. Kleinheins

Institut für Neutronenphysik und Reaktortechnik
Projekt Nukleare Sicherheitsforschung

April 1999

Forschungszentrum Karlsruhe
Technik und Umwelt

Wissenschaftliche Berichte

FZKA 5997

Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen

Teil 3: VISAPTER—ein Programm zum Anschluß einiger Visualisierungssysteme an VISART-Dateien

Siegfried Kleinheins

Institut für Neutronenphysik und Reaktortechnik
Projekt Nukleare Sicherheitsforschung

Forschungszentrum Karlsruhe GmbH, Karlsruhe
1999

Als Manuskript gedruckt

Forschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

Mitglied der Hermann von Helmholtz-Gemeinschaft
Deutscher Forschungszentren (HGF)

ISSN 0947-8620

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. Zur Visualisierung mit kommerziellen Visualisierungssystemen müssen die Daten aus dem VISART-Format in das von den Systemen erwartete Format umgesetzt werden. Das Programm VISAPTER liefert diese Umsetzung für die Systeme AVS/Express, Gsharp und Tecplot-7. Über die VISAPTER-Eingabe gesteuert, können dabei aus den VISART-Daten Teilmengen der Größen ausgewählt werden und für die Darstellung über dem gesamten Netz oder Querschnitten oder Ausschnitten daraus, auch in Kombination mit der Problemzeit oder allein über der Problemzeit, aufbereitet werden. Die von VISAPTER erstellten Ausgabedateien können von den Visualisierungssystemen, ggf. gesteuert über bereitgestellte Script-Dateien, unmittelbar eingelesen und visualisiert werden. Eine Sonderfunktion von VISAPTER erlaubt das Anschauen der ausgewählten Daten in Zahlenform.

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

Part 3: VISAPTER—a Program for Connecting some Visualization Systems to VISART Files

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. For visualizing by commercial visualization systems the data must be transformed from the VISART format into the format expected by the systems. The VISAPTER program provides the transformation for the AVS/Express, Gsharp and Tecplot-7 systems. Controlled by VISAPTER input, in this process subsets of the quantities from the VISART file can be selected and prepared for the visualization over the complete mesh or cross sections or sections of it, as well as in combination with the problem time or over the problem time alone. The output files generated by VISAPTER can directly be read in and visualized by the systems, controlled by supplied script files if so desired. A special feature of VIAPTER permits the numerical display of the selected data.

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 dieser 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 Dezember 1998) bereits in die folgenden, im FZK betreuten oder entwickelten Codes eingebaut: SIMMER-II.12, AFDM, SIMMER-III, HYDSOL, MATTINA, 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/Express, Gsharp, Tecplot-7), die jeweils eigene Formate für ihre Eingabedateien verlangen, 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/Express, Gsharp, Tecplot-7 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.

Inhalt

1. Einleitung

VISART ist der Name eines (zunächst im FZK-Bereich) standardisierten Formats für Postprocessor-Dateien [1b] mit über der Geometrie und der Zeit errechneten Daten, wie sie z.B. von Fluiddynamik-Codes zur anschließenden grafischen oder sonstigen Auswertung erstellt werden.

Im VISART-Konzept [1a] einer einheitlichen Schnittstelle zwischen Codes (Fluiddynamik- und verwandte Programme) und Auswerteprogrammen schreiben die Codes die errechneten auszuwertenden Daten auf eine Postprocessor-Datei im VISART-Format. Zur Auswertung der Daten mit einem kommerziellen Visualisierungssystem müssen sie zunächst in das vom jeweiligen System erwartete Eingabedateiformat umgesetzt werden. In diesem Prozess wird man in der Regel auch eine Auswahl aus den in der Datei enthaltenen Daten in Bezug auf Größen und Darstellungskoordinaten treffen.

Das zu diesem Zweck entwickelte Adapterprogramm VISAPTER[1] dient zum Anschluß der kommerziellen Visualisierungssysteme AVS/Express, Gsharp und Tecplot-7 (siehe letzter Absatz) an VISART-Dateien. Aus den auf den VISART-Dateien enthaltenen Daten kann, nach Maßgabe der Steuereingabe von VISAPTER, eine Teilmenge der Größen ausgewählt und ggf. ein Teilgebiet des Netzes und/oder ein Teilabschnitt der Problemzeit ausgeschnitten werden. Die so selektierten Daten werden dann für das jeweils gewünschte Visualisierungssystem aufbereitet und im von diesem geforderten Format auf eine Datei ausgegeben, die als Eingabe für das Visualisierungssystem dient (siehe Bild 1). Eine Sonderfunktion von VISAPTER erlaubt statt der Visualisierung die Anzeige der Zahlenwerte der ausgewählten Größen über dem ausgeschnittenen Teilgebiet und Teilabschnitt.

VISAPTER ist ein Fortran-Programm, das im Stapelbetrieb aufgerufen wird, und zwar am bequemsten über eine grafische Benutzeroberfläche, mit der die Steuereingabe zusammengestellt wird. Die grafische Benutzeroberfläche wurde bereits im Teil 1 der VISART-Dokumentation [1a] vorgestellt. Im vorliegenden Bericht, dem Teil 3 der VISART-Dokumentation, werden VISAPTER, seine Steuereingabe, seine Ein- und Ausgabedateien und die Anwendung für die genannten Visualisierungssysteme beschrieben. (Der Teil 3 ist eine völlig umgearbeitete und aktualisierte Neufassung eines unveröffentlichten Berichts vom Oktober 1994 und zweier späterer Ergänzungen dazu.)

Zum Verstehen des Berichts ist die Vertrautheit mit dem einführenden Teil 1 der Dokumentation [1a] Voraussetzung, besonders bezüglich der verwendeten Terminologie. Allerdings haben sich seit dessen Abfassung einige Änderungen ergeben, die besonders beim Vergleich der Tabellen und Bilder des Teils 1 und dieses Berichts auffallen. So sind die in [1a] zugrunde gelegten Visualisierungssysteme AVS, UNIGRAPH und Tecplot, Version 6, inzwischen durch ihre Nachfolge-Releases bzw. -Systeme AVS/Express, Gsharp und Tecplot, Version 7, ersetzt worden (daneben wurde der Code IVA-KA in MATTINA umbenannt). Eine Ausschöpfung der dadurch gegebenen Möglichkeiten, besonders auch für die Darstellung von dreidimensionalen Geometrien, sowie die Ausweitung der vom Benutzer wählbaren Darstellungsformen, bis hin zur Erzeugung von Bildsequenzen für die Animation oder Verfilmung, erforderten einige Änderungen und Erweiterungen im Programm VISAPTER und im zugehörigen Teil der VISART-Oberfläche, die nun hier auf aktuellem Stand dokumentiert sind.

[1] von „**VISA**RT-Ada**pter**“

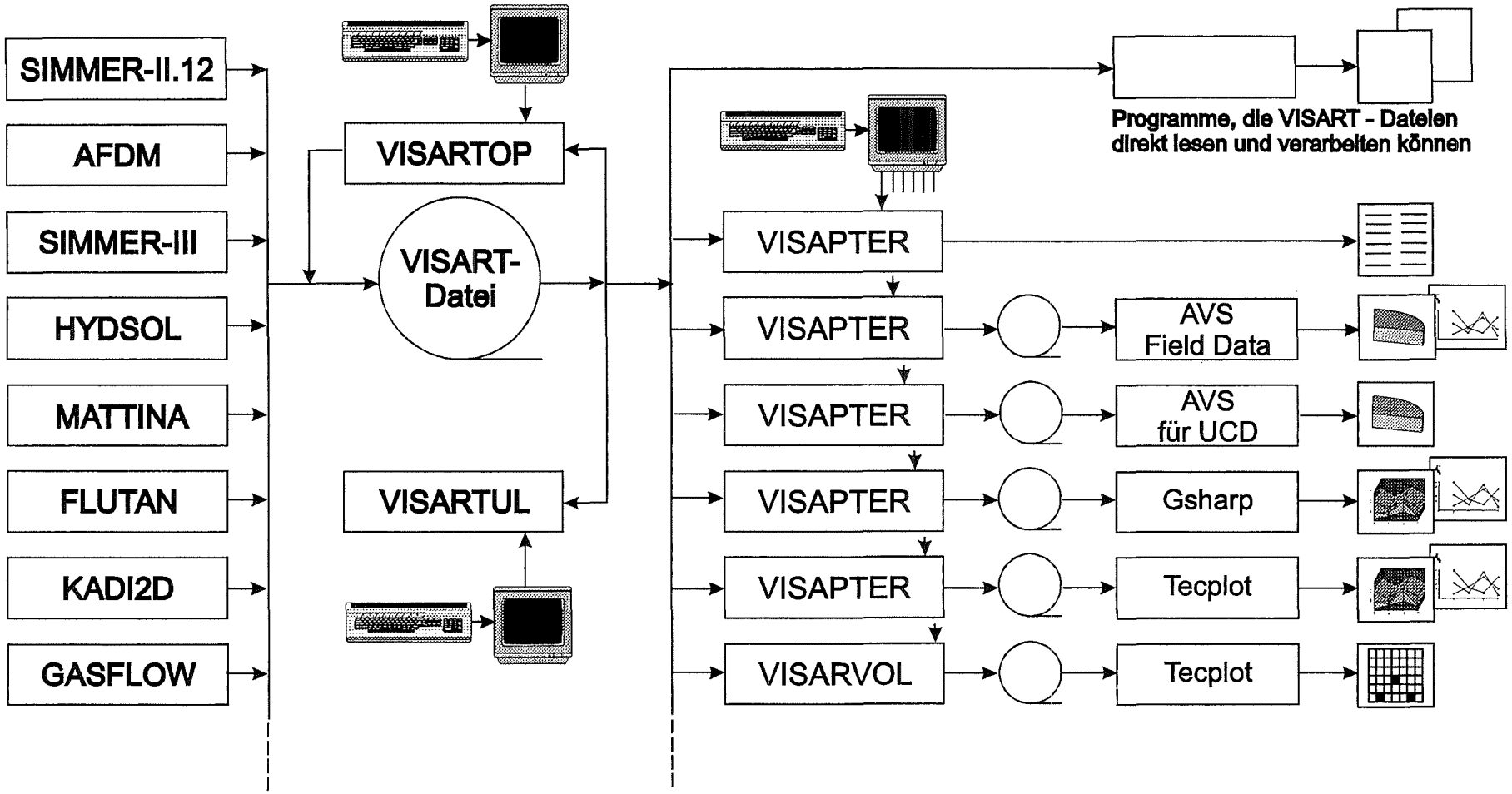


Bild 1 : Datenfluß über die VISART - Schnittstelle

2. Fähigkeiten des Programms

In diesem Kapitel werden die Visualisierungsoptionen des Programms VISAPTER vorgestellt.

2.1 Visualisierungssysteme

In der nachfolgenden Tabelle sind die kommerziellen Visualisierungssysteme, die über VISAPTER zur Zeit für VISART-Dateien zugänglich sind, mit einigen ihrer Fähigkeiten zusammengestellt. (Die Tabelle enthält auch einen Eintrag für die Sonderfunktion „Werteausgabe" von VISAPTER.)

Die Visualisierungssysteme beherrschen die folgenden in unserem Zusammenhang interessierenden Darstellungsformen:

- Alle (außer AVS-UCD) beherrschen konventielle Funktionsgrafik.
- Alle beherrschen die üblichen Repräsentationen skalarer und vektorieller Größen über 2D-Netzen in node-based (s.u.) Darstellung: Höhenlinien- oder Konturplots und perspektivische Plots von Skalarfeldern; Vektorplots für Vektorfelder.
- Alle erlauben Scatter-Plots.

Weitere Fähigkeiten sind der folgenden **Tabelle 2.1** zu entnehmen:

System	eff. Dim.	Regul. Netz kartesisch	Regul. Netz krummlinig	Irreg. Netz	Teilnetze	Leere Maschen (Löcher)	Defektive Netze	Cell-based Darst.	Festes Dateiformat
Werteausgabe	3	x	x	x	x	x	x		
AVS/Express AVS-FD[1] [3b]	3	x[(5)]	x[2]	x					x
AVS/Express AVS-UCD[1] [3b]	3	x	x[2]	x	x[3]	x[3]	x	x	x
Gsharp Release 2.1 [4a]	3	x[5]	(x[6])			x	x[4]	x	
Tecplot Version 7 [5a]	3	x	x[2]	x	x	x	x[4]	x	x

(Weitere 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
[6] nur Polarkoordinaten, in Form von Koordinaten pro Maschenindex
(x): noch nicht implementiert

Erklärungen:

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

- Kartesische und krummlinige Koordinaten regulärer Netze können im Prinzip sowohl als Koordinaten pro Koordinatenrichtung, als auch als Koordinatenpaare/tripel pro Masche/Punkt an das System übergeben werden. Kartesische Koordinaten werden bei AVS-FD und Gsharp als Koordinaten pro Koordinatenrichtung übergeben. Krummlinige Koordinaten werden bei den aufgeführten Visualisierungssystemen (außer bei Gsharp) stets als Koordinatenpaare/tripel pro Masche/Punkt übergeben. Das gleiche gilt für irreguläre Netze.

- Netze, die nicht voll belegt sind, können im Prinzip aus Teilnetzen zusammengesetzt sein, durch Markierung der leeren Maschen („Löcher") definiert sein, oder als „defektive Netze" überhaupt nur aus den explizit aufgezählten Maschen bestehen (siehe [1a, 1b]). Teilnetze sind nur in Tecplot (als „Zonen") direkt definierbar. Leere Maschen sind in Gsharp (mit Fragezeichen als „undefined values") und Tecplot (über eine „blanking" Variable) definierbar. Defektive Netze lassen sich direkt nur mit AVS-UCD darstellen. Mit VISAPTER lassen sich auf den in der Tabelle markierten Umwegen die verschiedenen Realisierungen nicht-voller Netze aber auch mit den anderen Systemen darstellen.

- 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. der Maschenmitte (siehe 2.3). Zur Zeit gibt es cell-based Darstellungen nur für skalare Größen (mit „cell-based Modules" in AVS-UCD; als „Grid graph" in Gsharp; als „Contour Plot" mit „Corner Flooding" in Tecplot).

- „Festes Dateiformat" bezieht sich auf das Format, in dem das System die zu visualisierenden Daten erwartet (siehe 6.2.2, 6.3.2, 6.4.2). Die nicht in dieser Spalte aufgeführten Systeme können die Daten auch lesen, sofern sie lediglich in einer tabellarischen Form angeliefert werden.

2.2 Darstellungskoordinaten

Soweit vom Visualisierungssystem her möglich, können mit VISAPTER Geometrien verarbeitet werden, die auf regulären oder irregulären, vollen oder nicht-vollen, **Netzen** (siehe [1a, 1b]) beruhen.

Zur Visualisierung von Netzgrößen (Gruppen 5/15) und Maschengrößen (Subgruppen 7/17) **über der Geometrie** kann VISAPTER, je nach Steuereingabe, aus allen diesen Netzen Teilgebiete ausschneiden und damit eindimensionale Darstellungen (Funktionsgraphen entlang einer Netz-Traverse), zweidimensionale Darstellungen (die üblichen Plots über einer Netzebene), oder—bei dreidimensionalen Netzen—auch dreidimensionale Darstellungen veranlassen.

Teilgebiete sind hier analog zu den Teilnetzen (siehe [1a, 1b]) definiert. Während Teilnetze zur Zusammensetzung eines Netzes in der VISART-Datei gebraucht werden, dienen Teilgebiete zur Zerlegung des Netzes für die Visualisierung. Ein **Teilgebiet** ist ein Ausschnitt aus einem (vollen oder nicht-vollen) Netz, der durch Festlegung von Anfangs- und Endmaschenindizes innerhalb des Netzes in einer oder mehreren Koordinatenrichtungen entsteht. Fallen Anfangs- und Endmaschenindex eines Teilgebiets in einer oder mehreren Koordinatenrichtungen zusammen, so sprechen wir von einem degenerierten Teilgebiet. Je nach Darstellungsweise[1] (siehe 2.3) läßt sich ein degeneriertes Teilgebiet auch als ein **Subgebiet** auffassen. Ein Subgebiet entsteht durch einen Querschnitt (oder Querschnitte) durch das Netz, d.h. durch Konstanthalten einer oder mehrerer Koordinaten des Systems. Ein Subgebiet hat deshalb eine „Subdimension" kleiner als die Netzdimension (im Extremfall die Subdimension Null)[1].

Die Netzkoordinaten des ausgewählten Teilgebiets können mit VISAPTER ggf. bearbeitet werden. Netze mit krummlinigen Koordinaten lassen sich, wenn dies vom Visualisierungssystem her erforderlich ist, für die Darstellung auf kartesische Koordinaten **umrechnen**. 2-dimensionale ϑ, z-Netze/Teilgebiete/Subgebiete lassen sich auf eine Ebene **abwickeln**. 2-dimensionale r, z-Netze/Teilgebiete lassen sich an der z-Achse **spiegeln**.

Neben Darstellungen über der Geometrie zu festen Problemzeitpunkten kann VISAPTER auch Größen der Gruppen 15 oder der Subgruppen 17 an spezifizierten Maschen, oder Größen der Gruppen 19 oder Subgruppen 21 schlechthin, für die Darstellung **über der Problemzeit** (durch Funktionsgraphen) aufbereiten. Hierzu kann VISAPTER, je nach Steuereingabe, auch Teilintervalle der Problemzeit auswählen.

Daneben ist auch eine mehrdimensionale Darstellung **über der Geometrie und der Problemzeit** möglich: 1- oder 2-dimensionale Netze/Teilgebiete/Subgebiete lassen sich mit VISAPTER durch Hinzunahme der (eventuell skalierten) Zeitkoordinate auf eine 2- bzw. 3-dimensionale effektive Dimension für die Visualisierung der Größen erweitern (dies geht nur in node-based Darstellung—siehe 2.3). (Bei einigen Visualisierungssystemen (siehe Tab. 3.2.1) lassen sich die Werte der Zeitkoordinate für die Darstellung durch Nullen ersetzen. Damit verbleibt eine „degenerierte", aber mehrfache Darstellung nur über der Geometrie, die im Visualisierungssystem ggf. in eine zeitliche Folge von Darstellungen über der Geometrie umgewandelt werden kann (siehe 3.2).)

[1] In cell-based Darstellung (siehe 2.3) kann ein degeneriertes Teilgebiet auch mit der vollen Netzdimension dargestellt werden.

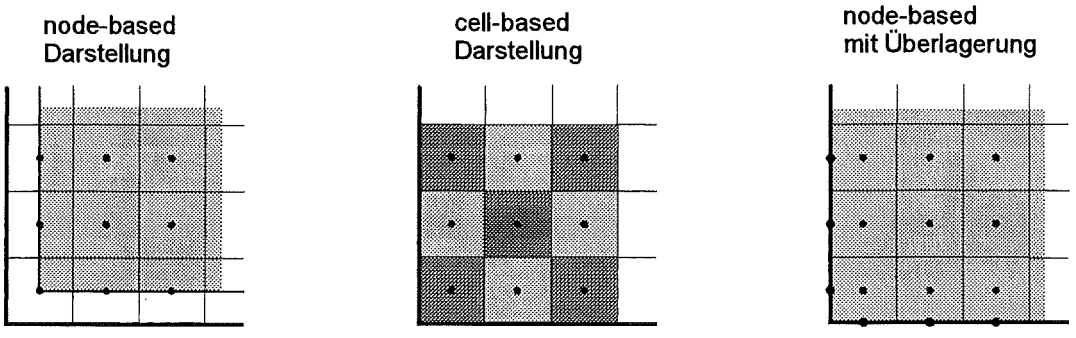
2.3 Darstellungsweisen

Den Ort, in dem eine Größe der Gruppen 5/15 oder der Subgruppen 7/17 in Bezug auf die Maschen dargestellt werden soll, entnimmt VISAPTER der in der Gruppe/Subgruppe stehenden Lokationsangabe [1a, 1b]. Bei der 2D- oder 3D-Visualisierung von Größen in der üblichen sog. **„node-based"** Darstellung werden deren Werte vom Visualisierungssystem nur in dem Gebiet wiedergegeben, das durch Verbinden der jeweiligen Lokationen („Nodes") eingeschlossen wird. Falls die Größen nur über Lokationen in den Maschenmitten (Lokationsindikator 0) oder über Lokationen auf den Maschenhüllen (Lokationsindikator 11, 22, 44) gegeben sind, geben die Verbindungslinien dieser Lokationen nicht das Gitter des Netzes wieder. Auch fehlt an der Netzhülle eine halbe Maschenbreite; die Geometrie des Netzes wird nur unvollständig dargestellt. Lediglich wenn die Lokationen außer den Maschenmitten auch die Punkte auf der Netzhülle einschließen (Lokationsindikator 88), oder überhaupt nur Gitterpunkte umfassen (Lokationsindikator 99), liefert die node-based Darstellung die vollständige Geometrie des Netzes.

Eine geometrisch vollständige Darstellung des Netzes ist mit VISAPTER aber noch auf andere Weise möglich (siehe die untenstehenden Bilder):

Bei Lokationen nur in den Maschenmitten[1] kann VISAPTER wahlweise die sog. **„cell-based"** Darstellung veranlassen, soweit diese vom Visualisierungssystem unterstützt wird (siehe 2.1). Dabei wird jeweils der ganzen Masche (anstelle nur dem Maschenmittelpunkt) der Wert der Größe zugeordnet, z.B. durch einheitliche Färbung (eine Interpolation zwischen den Maschen findet nicht statt). Die „Cells" dieser Darstellungsweise entsprechen den Maschen; das Netzgitter und die Netzgeometrie werden richtig dargestellt. Cell-based Darstellung ist nur über der Geometrie allein möglich, nicht in Kombination mit der Problemzeit.

Wenn die Werte von Größen über Lokationen auf der Netzhülle nicht zusammen mit den Werten im Netzinneren gespeichert sind (Lokationsindikator 88), sondern in eigenen Gruppen/Subgruppen (Lokationsindikator -11, -22, -33, -44, -77, -88), kann VISAPTER sie mit den Werten der Größen aus den Gruppen/Subgruppen für die Maschenmitten (Lokationsindikator 0) durch **Überlagerung** so kombinieren, daß die Geometrie des Netzes bei der Visualisierung in node-based Darstellung auch hier vollständig wiedergegeben wird. Diese separate Speicherung der Werte auf der Netzhülle empfiehlt sich z.B., wenn es sich um konstante Randwerte handelt, die nur im Kopf-Paket gespeichert werden müssen, oder sie ist unumgänglich bei defektiven Netzen (Subgruppen 7/17), wo die Werte auf der Netzhülle nicht zusammen mit den Werten im Netzinnern gespeichert werden können, weil hier keine eindeutige Indexzuordnung möglich ist. (Im letzteren Fall ist, zusätzlich zu den Subgruppen 7/17 mit den Maschengrößen, eine Subgruppe 7/17 mit dem Oberflächennormalenvektor des Netzes in der VISART-Datei (siehe 4.1) erforderlich. Auf diese Weise sind auch innere Netzhüllen—Hohlräume und innere Wände, in einfacher Struktur—darstellbar, allerdings bisher nur mit AVS-UCD.) VISAPTER inter- oder extrapoliert die durch die vorhandenen Lokationen nicht abgedeckten „Nodes", z.B. auf Ecken der Netzhülle.



[1] Im Falle zweidimensionaler Subgebiete durch die Maschenhüllen dreidimensionaler Netze: auch bei Lokationen in den Mitten der Maschenhüllen (Lokationsindikatoren 11, 22, 44).

Die folgende **Tabelle 2.3.1** zeigt die möglichen Lokationen zur Darstellung des **Netzinnern**, ggf. auch einschließlich der Netzhülle, mit den Visualisierungssystemen:

	Gruppen 5/15				Subgruppen 7/17			
Lokation	0	11,22,44	88	99	0	11,22,44	33/77 + Normale	99
AVS-FD	x	x	x	x				
AVS-UCD	x	x	x	x	x	x	(x^2)	x
Gsharp	x	x	x	x	x^4	x^4		x^4
Tecplot	x	x	x	x	x^4	x^4	(x^2)	x^4

Die folgende **Tabelle 2.3.2** zeigt die möglichen Lokationen zur Darstellung der **Netzhülle** allein mit den Visualisierungssystemen:

	Gruppen 5/15				Subgruppen 7/17			
Lokation						-11,-22,-44	-33/-77 + Normale	
AVS-FD								
AVS-UCD						x	x	
Gsharp						x^4	x^4	
Tecplot						x^4	x^4	

Die folgende **Tabelle 2.3.3** zeigt die möglichen Darstellungsweisen (cell-based oder node-based mit Überlagerung von Randwerten) des **Netzinnern zusammen mit der Netzhülle** mit den Visualisierungssystemen. Für node-based Darstellung sind die hier möglichen Lokationen zur Darstellung der Netzhülle angegeben; in beiden Darstellungsweisen muß hier das Netzinnere an der Lokation 0 definiert sein.

	Gruppen 5/15			Subgruppen 7/17		
	cell-	node-based		cell-	node-based	
Lokation		-11,-22,-44	-88		-11,-22,-44	-33/-77 + Normale
AVS-FD		(x)	x			
AVS-UCD	x	(x)	x	x	x	x
Gsharp	x	(x)	x	x^4		
Tecplot	x	(x)	x	x^4	(x^2)	(x^2)

(x): noch nicht implementiert
[2] prinzipiell möglich, aber nicht implementiert
[4] durch Expansion auf Gruppen 5/15 mit Löchern

Falls Werte von Größen über Lokationen auf der Netzhülle in eigenen Gruppen/Subgruppen gespeichert sind (Lokationsindikator -11, -22, -33, -44, -77, -88), kann VISAPTER auch die **Oberfläche** des Netzes allein darstellen. Dies geht natürlich nur auf Subgebieten, die mit der Oberfläche (zumindest teilweise) zusammenfallen.

In den Abschnitten 2.2 und 2.3 bezogen wir uns bisher auf die Darstellung von Netz- und Maschengrößen, die Skalar- und Vektorfelder beschreiben. Für Punktgrößen der Gruppen 6/16, die Partikel(sorten) beschreiben, kann VISAPTER die Darstellung als **Scatter-Plots** mit den Visualisierungssystemen vorbereiten. Die Gruppen 6/16 definieren die Koordinaten der Partikel, die (als Gruppen 16) auch zeitlich variabel sein können. Dies genügt bereits als Mindestangabe für die Darstellung eines Scatter-Plot. Weitere Eigenschaften der Partikel(sorten), z.B. Durchmesser oder Geschwindigkeit der Partikel, können in den zur Gruppe 6/16 gehörenden Subgruppen 7/17 stehen, und werden von VISAPTER an das Visualisierungssystem zur Interpretation übergeben.

3. Steuerung des Programms

In diesem Kapitel wird die Steuereingabe des Programms VISAPTER beschrieben.

3.1 Steuereingabe

Das Programm ist für den Stapelbetrieb entworfen; es erwartet eine komplette Steuereingabe auf einer Datei und läuft vom Start bis zum Programmende oder Fehlerabbruch ohne Wechselwirkung mit dem Benutzer.

Hier wird der Aufbau der Steuereingabe beschrieben, wie sie vom Benutzer zu erstellen ist bzw. wie sie von der grafischen Benutzeroberfläche (siehe 3.4) aus der Einstellung der „Widgets“ erzeugt wird.

In der Beschreibung der Eingabe entspricht jede Zeile *einem* (logischen) Satz der Steuereingabe. Die Eingabe wird listengesteuert gelesen; mit `I,...,N` beginnende Namen bezeichnen `INTEGER`-Variable, mit `C` beginnende Namen bezeichnen `CHARACTER`-Variable aus bis zu 8 Zeichen, `FORMT` bezeichnet eine `CHARACTER`-Variable aus bis zu 72 Zeichen, alle anderen Namen bezeichnen `REAL`-Variable.

1. `MODE IZ IKRUM KNULL IV IRAND IRAKD`

Wenn `IV` $\neq 0$: eine Zeile 2:

2. `CC(1) ... CC(|IV|) CNORM`

Wenn `KNULL` > 0 : eine Zeile 3:

3. `FORMT`

Wenn `IZ` > 0 : eine Zeile 4:

4. `TX`

(Das folgende Zeilenarrangement 5 bis 7 wird für jede VISART-Datei—siehe 4.1—wiederholt:)

5. `NF NP NT NN`

Wenn `NT` $\neq 0$ ($\wedge$ `|NT|` ≤ 20) : eine Zeile 6:

6. `T1(1) T2(1)) ... T1(|NT|) T2(|NT|)`

Wenn `NN` > 0 : `NN` Zeilen 7':

7'. `NG CN CG IKOM KOR1(1) KOR2(1) KOR1(2) KOR2(2) KOR1(3) KOR2(3)`

Wenn `NN` < 0 : `|NN|` Zeilen 7":

7". `CI`

Das Programm VISAPTER erwartet die Steuereingabe als Standardeingabe. Wenn die Steuereingabe direkt mit der Tastatur eingegeben wird, springt das Programm nach dem Verarbeiten der letzten Eingabezeile wieder auf die Zeile 5 und erwartet Eingabe. Dort ist dann die Zeile

```
-1 0 0 0
```

für das Eingabeende einzutippen. Wenn die Steuereingabe von einer umgelenkten Datei kommt, wird das Dateiende (EOF) als Ende der Eingabe erkannt.

Bedeutung der Eingabeparameter:

`MODE` wählt das Visualisierungssystem, für das die Ausgabe erstellt werden soll:
- 0: keine Visualisierung, sondern Inspektion der Werte (siehe 6.1);
- 1: AVS mit Field-Data (FD) (nur node-based, siehe 6.2);
- +3: Gsharp, node-based (siehe 6.3);
- −3: Gsharp, cell-based (siehe 6.3);
- +4: AVS mit Unstructured-Cell-Data (UCD), node-based (siehe 6.2);
- −4: AVS mit Unstructured-Cell-Data (UCD), cell-based (siehe 6.2);
- +5: Tecplot, node-based (siehe 6.4);
- −5: Tecplot, cell-based (siehe 6.4).

`IZ` gibt an, wie die effektive Dimension der Visualisierungskoordinaten aus den räumlichen und zeitlichen Dimensionen der Teilgebiete und Problemzeitabschnitte zusammengesetzt wird:
- −9: räumliche Dimensionen gemäß Teilgebietauswahl;
- −3: 3 räumliche Dimensionen;
- −2: 2 räumliche Dimensionen;
- −1: 1 räumliche Dimension;
- 0: nur zeitliche Dimension;
- 1: 1 räumliche und 1 zeitliche Dimension;
- 2: 2 räumliche und 1 zeitliche Dimensionen;
- 3: 3 räumliche und 1 zeitliche Dimensionen;
- 9: räumliche Dimensionen gemäß Teilgebietauswahl und 1 zeitliche Dimension.

`IKRUM` wirkt nur bei regulären Netzen und gibt bei krummlinigen Koordinaten die Art der Koordinatenumrechnung bzw. bei r,z-Netzen/Teilgebieten die Spiegelungsoption an:
- 0: keine Umrechnung bzw. keine Spiegelung;
- 1: Umrechnung der Vektorkomponenten und Koordinaten auf kartesische Koordinaten;
- 2: bei Zylindermantel und Kreislinie: Umrechnung der Koordinaten auf die (kartesischen) Koordinaten der Abwicklung;
- −1: Spiegelung des rechteckigen r,z-Netzes/Teilgebiets an der z-Achse.

`KNULL`
- > 0: gibt für `IZ` $\neq 0$ die Nummer n derjenigen Koordinate des Teilgebiets (in der Reihenfolge i,j,k bzw. i,j,t bzw. i,t) an, deren Zeit- bzw. Koordinatenwerte in der Ausgabedatei durch Nullen ersetzt und stattdessen auf eine eigene Datei Nr. 22 ausgegeben werden sollen (siehe 3.2);
- < 0: gibt an, daß ein- oder zweidimensionale Subgebiete im dreidimensionalen Raum dargestellt werden sollen, wobei `|KNULL|` die Nummer n bzw. die Nummern $n_1 n_2$ derjenigen Raumkoordinate(n) (in der Reihenfolge i,j,k) angibt, die auf der Subgebietsfläche senkrecht steht bzw. die auf der Subgebietslinie senkrecht stehen (siehe 3.2);
- 0: wenn keine Koordinatenwerte durch Nullen ersetzt bzw. separat ausgegeben und 2D-Subgebiete nicht im 3D-Raum dargestellt werden sollen.

IV — steuert die Überlagerung mit Größen auf der Netzhülle bzw. die Darstellung auf der Oberfläche allein von Größen auf der Netzhülle:
0: keine Überlagerung/Oberflächendarstellung;
±1: Oberflächendarstellung von Größen auf der Netzhülle;
±2, ...: Überlagerung von |IV|−1 Größen auf der Netzhülle mit der Größe im Netzinneren;
sign(IV) gibt, falls CNORM nicht leer ist, die Richtung des Oberflächennormalenvektors (siehe 4.1) für die Netzhülle an (+1: Normale zeigt ins Netzäußere; −1: Normale zeigt ins Netzinnere).

IRAND — gibt an, wo (d.h. nach dem Lesen welcher Pakete) VISAPTER die Zellstruktur oder Überlagerungsstruktur des Teilgebiets aus den Gruppen der auszugebenden Größen anlegen soll (siehe 5.4):
> 3: nur in dem Paket, wo die entsprechenden Gruppen zuerst gefunden werden;
3: nur im Kopf-Paket des ersten Files;
2: in jedem Kopf-Paket;
1: nur im ersten Rumpf-Paket;
0: in jedem Rumpf-Paket;
-1: überall, wo die entsprechenden Gruppen gefunden werden.

IRAKD — gibt an, wo (d.h. nach dem Lesen welcher Pakete) VISAPTER die Koordinaten des Netzes aus den Gruppen mit den Netzkoordinaten entnehmen soll (siehe 5.4):
> 3: nur in dem Paket, wo die entsprechenden Gruppen zuerst gefunden werden;
3: nur im Kopf-Paket des ersten Files;
2: in jedem Kopf-Paket;
1: nur im ersten Rumpf-Paket;
0: in jedem Rumpf-Paket;
-1: überall, wo die entsprechenden Gruppen gefunden werden.

(für |IV| = 1:)

CC(1) — Ziffer mit Vorzeichen (als Character-Konstante!); die Ziffer bezeichnet die Nummer der Koordinate des Netzes, in die die Normale des darzustellenden Subgebiets der Netzhülle zeigt; das Vorzeichen bezeichnet bei inneren Wänden die jeweils zu berücksichtigende Seite der Netzhülle (genau 2 Zeichen).

(für |IV| = 2,...:)

CC(i) — (i=1,...,|IV|:)
Prefix oder Suffix der Identifikationen der Größen im Inneren des Netzes (für i=1) oder der (i-1)ten zu überlagernden Größen auf der Netzhülle (siehe 4.1) (maximal 8 Zeichen; mindestens das letzte bzw. erste Zeichen muß leer sein);

CNORM — Identifikation des Oberflächennormalenvektors der Netzhülle, falls vorhanden (siehe 4.1); sonst: leer.

FORMT — Character-Format-Spezifikation für die Ausgabe der Zeit- bzw. Koordinatenwerte auf Datei Nr. 22 (siehe KNULL > 0 und 3.2).

TX — Faktor, mit dem für IZ > 0 die Problemzeitkoordinate multipliziert wird, um ihre Zahlenwerte an die der geometrischen Koordinaten anzupassen.

NF
> 0: Anzahl der zu verkettenden Fortsetzungs-Files der VISART-Datei (siehe 4.1);
−1: für das Ende der Steuereingabe (s.u.).

NP
- ≥ 0: Maximale Anzahl der auszugebenden Rumpf-Pakete (falls IZ < 0);
- < 0: die Rumpf-Pakete, deren Zyklusnummern Vielfache von |NP| sind, werden ausgegeben (falls IZ < 0).

NT
- 0: Alle in der Datei enthaltene Rumpf-Pakete werden (nach Maßgabe von NP, falls IZ < 0) ausgegeben;
- $\neq 0 \wedge$ |NT| ≤ 20 : nur die Rumpf-Pakete in den durch T1, T2 spezifizierten Problemzeitintervallen werden (nach Maßgabe von NP, falls IZ < 0) ausgegeben;
- $> 20 \vee < -20$: nur das letzte Rumpf-Paket in der Datei wird ausgegeben.

NN
- 0: Alle in den Rumpf-Paketen enthaltenen Größen der Gruppe 15, wenn IZ $\neq 0$, oder der Subgruppe 21, wenn IZ $= 0$, werden ausgegeben;
- > 0: nur die durch Zeile 7' spezifizierten Größen werden ausgegeben;
- < 0: nur die in den Rumpf-Paketen enthaltenen Größen der Gruppe 15, wenn IZ $\neq 0$, oder der Subgruppe 21, wenn IZ $= 0$, die in Zeile 7" *nicht* spezifiziert sind, werden ausgegeben.

T1(m), T2(m) (m=1,...,|NT|:) spezifizieren die ausgewählten Problemzeitintervalle:
- bei NT > 0 : T1(m) und T2(m) begrenzen das m-te Intervall;
- bei NT < 0 : T1(m) und T2(m) definieren eine gleichmäßige Folge von Intervallen (siehe 3.2).

NG Gruppen-Kennzahl der auszugebenden Größe:
- 5/15: Netzgröße;
- 6/16: Punktgröße (Partikelsorte);
- 7/17: Maschengröße;
- 19: Integralgröße;
- 21: Zeitfunktionsgröße.

CN
- bei NG = 5/15, 7/17 oder 21 : Identifikation der auszugebenden Größe;
- bei NG = 6/16 : Identifikation der Partikelsorte;
- bei NG = 19 : leer oder der Name der Größe in der beschreibenden Integralkonstanten-Gruppe mit der Identifikation CG.

CG
- bei NG = 6/16 : gleich CN;
- bei NG = 7/17 : Identifikation der Referenzgruppe 6 oder 16, auf die sich die Subgruppe für CN bezieht;
- bei NG = 19 : Identifikation der Integralgrößen-Gruppe, die die auszugebende Größe enthält, gegeben durch ihre Position KOR1(1) in der Gruppe oder durch die Position ihres Namens CN in der beschreibenden Integralkonstanten-Gruppe 9 im Kopf-Paket; im letzteren Fall gleichzeitig Identifikation der beschreibenden Integralkonstanten-Gruppe (siehe 3.2);
- andernfalls: leer.

IKOM gibt die auszugebenden Vektorkomponenten an:
- 0: wenn eine skalare Größe oder alle Vektorkomponenten einer vektoriellen Größe ausgegeben werden sollen;
- > 0: wenn die Vektorkomponente IKOM einer vektoriellen Größe ausgegeben werden soll;
- < 0: wenn bis zu |IKOM| der im jeweiligen Netz/Teilgebiet liegenden Vektorkomponenten einer vektoriellen Größe ausgegeben werden sollen.

(für NG = 5/15 oder = 7/17:)

KOR1(n), KOR2(n) ($n = 1, 2, 3$:) geben die Maschenindizes jeder Koordinatenrichtung (i, j, k) für die untere bzw. obere Grenze des auszugebenden Teilgebiets oder der Subgebietsscharen an:

$-1, -1$: wenn keine Grenzen gesetzt sind oder die Koordinate fehlt;
$N, -1$: wenn beide Grenzen auf N zusammenfallen;
N, M: wenn die Grenzen $N \leq M$ gesetzt werden.

(für NG = 19:)

KOR1(1) wenn CN leer ist: gibt die Position der auszugebenden Größe in der Integralgrößen-Gruppe CG an:

-1 oder 1: für die einzige oder erste Größe in der Gruppe;
N: für die N-te Größe in der Gruppe;
0: für alle Größen in der Gruppe;

andernfalls: gibt die laufende Nummer zum Namen CN der Größe in der beschreibenden Integralkonstanten-Gruppe mit der Identifikation CG an:

-1 oder 1: für das einzige oder erste Auftreten des Namens in der Gruppe;
N: für das N-te Auftreten des Namens in der Gruppe;
0: für jedes Auftreten des Namens in der Gruppe.

KOR2(1) -1

KOR1(2), KOR2(2), KOR1(3), KOR2(3): -1

(für NG = 21:)

KOR1(n) ($n = 1, 2, 3$:)
bei Netz- oder Maschengrößen: gibt die Maschenindizes jeder Koordinatenrichtung (i, j, k) für die spezifizierte Maschengröße an:

N: Maschenindex;
-1: wenn die Koordinate nicht vorhanden ist;

bei Integralgrößen: -1

KOR2(n) ($n = 1, 2, 3$:) -1

CI Identifikation der *nicht* auszugebenden Größe.

3.2 Anmerkungen zu den Eingabeparametern

|IZ|—erhöht um 1, falls IZ ≥ 0—gibt die effektive Dimension des Darstellungskoordinatensystems für die Visualisierung an. Die Dimension 1 eignet sich z.B. zur Darstellung durch Funktionsgraphen, die Dimension 2 zur Darstellung durch Kontur- und perspektivische Plots, die Dimension 3 zur dreidimensionalen Visualisierung.

Im einfachsten Fall werden für IZ < 0 die Größen nur für *einen* Problemzeitpunkt ausgegeben. Wenn es von der Visualisierung her möglich und sinnvoll ist (z.B. für mehrere Funktionsgraphen zu unterschiedlichen Problemzeiten in *einem* Bild—siehe Beispiele in 6.4.4—oder auch für eine Verfilmung von 2- oder 3dimensionalen Bildern), darf auch bei IZ < 0 mehr als nur ein Problemzeitpunkt (über NP, NT, ...) spezifiziert werden.

Die Angabe der Dimension des Darstellungskoordinatensystems durch IZ hat (außer bei |IZ| $= 9$) Vorrang über die Spezifikationen durch die anderen Eingabeparameter. Falls diese inkompatibel mit IZ sind, wird der Lauf mit einer Fehlermeldung abgebrochen.

IKRUM wirkt nur für reguläre Netze:

- IKRUM > 0 wirkt nur für Größen der Gruppe 15 und Subgruppe 17 und bezieht sich immer auf das Koordinatensystem des jeweiligen Netzes oder Teilgebiets der auszugebenden Größe[1].
  Bei IKRUM $= 1$ (Koordinatenumrechnung) und krummlinigen Netzen/Teilgebieten/Subgebieten oder Einzelmaschen aus solchen werden die Vektorkomponenten und Koordinaten ggf. auf kartesische Koordinaten umgerechnet.
  Bei IKRUM $= 2$ (Abwicklung) und einem (abwickelbaren) ϑ, z-Teilgebiet/Subgebiet oder Einzelmaschen daraus wird die Koordinate ϑ auf die Koordinate u der Abwicklung umgerechnet ($u = \rho_0\vartheta$, wo ρ_0 die festgelegte Koordinate des Teilgebiets/Subgebiets ist); die Reihenfolge der Vektorkomponenten (falls vorhanden) ist dann ρ, u, z.
- IKRUM < 0 (Spiegelung) wirkt nur für Größen der Gruppe 15 (aber nicht bei Teilnetzen und—außer für AVS-UCD—nicht bei defektiven Netzen) und bezieht sich immer auf das Koordinatensystem des jeweiligen Netzes oder Teilgebiets der auszugebenden Größe[1]. Hier wird bei einem 2dimensionalen (ebenen, rechteckigen) r, z-Netz/Teilgebiet (das voll an die z-Achse anschließen muß) das Netz/Teilgebiet um das an der z-Achse gespiegelte Netz/Teilgebiet ergänzt; das Vorzeichen der Vektorkomponenten in r-Richtung und der r-Koordinatenwert kehren sich in der gespiegelten Ebene um.[2]

KNULL > 0 kann nur solche Koordinaten spezifizieren, deren komplementäre Koordinaten ein ebenes bzw. gerades Teilgebiet aufspannen. Bei MODE $= \pm 4$ kann KNULL $\neq 0$ nur die Koordinate t bezeichnen. Die Möglichkeit, die in der Spezifizierung von KNULL liegt, erlaubt z.B. die Erstellung einer Folge von zweidimensionalen-Darstellungen für die Verfilmung auch dann, wenn das Visualisierungssystem nur *eine* Darstellung pro Aufruf vorsieht (d.h. wenn IZ < 0 nur mit *einem* Problemzeitpunkt erlaubt ist), indem die Zeitachse als 3. Koordinate hinzugenommen wird (IZ > 0), die dann mit KNULL in der Darstellung unterdrückt wird und nur zur Auswahl der jeweiligen Zeitebene bei der Weiterverarbeitung im Visualisierungssystem dient. Da in diesem Fall die unterdrückten Werte der Zeitachse mit dem Format FORMT auf die Datei Nr. 22 geschrieben werden, können diese als Labels für die einzelnen Filmbilder dienen.

KNULL < 0 wird benötigt, wenn ein- oder zweidimensionale Subgebiete im dreidimensionalen Raum dargestellt werden sollen. Für eindimensionale Subgebieten ist |KNULL| $= n_1 n_2$, für zweidimensionale Subgebieten ist |KNULL| $= n$. KOR1(n) und KOR2(n) (und ebenso KOR1(n_1), KOR2(n_1) KOR1(n_2), KOR2(n_2)) bezeichnen den konstantzuhaltenden Index (bzw. die konstantzuhaltenden Indizes) für das Subgebiet oder den Indexbereich für die Subgebietsschar.

FORMT könnte z.B. den Wert (''t = '',F6.4,'' s'') haben (bei der listengesteuerten Eingabe muß diese Zeichenkette in Apostrophe eingeschlossen werden).

---

[1] Man beachte, daß Vektorkomponenten in Richtung der Koordinatenlinien und in Einheiten bezogen auf die Länge (nicht auf den Winkel!) in der VISART-Datei stehen.

[2] Spiegelung ist nicht möglich für Größen mit Lokation 01.

NP ist nur für IZ < 0 relevant.

Für IZ ≥ 0 muß NT $= 0$ oder NT $= 1$ sein. Wenn NT $= 0$ ist, werden alle Rumpf-Pakete ausgegeben. Wenn NT $= 1$ ist, werden alle der im durch T1(1), T2(1) spezifizierten Problemzeitrahmen liegenden Rumpf-Pakete ausgegeben.

Für IZ < 0 kann NT alle Werte einnehmen. Im Falle NT < 0 sind die Problemzeitintervalle durch

$$\mathtt{T2}(m-1) + n \times \mathtt{T1}(m) \pm 0.1 \times \mathtt{T1}(m)$$

gegeben, wo $m = 1, \ldots, |\mathtt{NT}|$ ist; $\mathtt{T2}(0) = 0$ und $n = 0, 1, \ldots$, solange wie $\mathtt{T2}(m-1) + n \times \mathtt{T1}(m) < \mathtt{T2}(m)$ bleibt. Falls mehrere Rumpf-Pakete in einem Problemzeitintervall liegen, wird nur das erste davon ausgewählt.

Für NN > 1 müssen die durch die Zeile 7' spezifizierten Größen in der Teilgebietsdimension oder -lage, in den Lokationen, im Typ, oder in anderen Eigenschaften, je nach der spezifischen Anwendung (siehe 6.), zueinander passen. (Andernfalls wird der Lauf mit einer Fehlermeldung abgebrochen. Bei NN ≤ 0 wird der Lauf *nicht* abgebrochen, wenn eine der auf der Postprocessor-Datei angetroffenen Größen in den obigen Eigenschaften nicht zu den vorher angetroffenen Größen paßt, sondern es wird lediglich eine Warnung gemeldet und die Größe nicht weiter verarbeitet.)

Im einfachsten Fall werden die Größen nur für *ein* Teilgebiet (definiert durch KOR1 und KOR2) ausgegeben. Wenn es von der Visualisierung her möglich und sinnvoll ist (z.B. für mehrere Funktionsgraphen von Werten an unterschiedlichen Maschen über der Problemzeit in *einem* Bild—siehe Beispiele in 6.4.4), darf die gleiche Größe auch mehrfach, mit unterschiedlicher Teilgebietsangabe, spezifiziert werden.

Die Eingabeparameter KOR1 und KOR2 beziehen sich auch bei degenerierten Netzen auf die Netzdimension und die in der jeweiligen Gruppe angegebenen Lokationen (Maschenmittelpunkte oder Maschenhüllen). Bei Lokationen in der Maschenmitte beginnt die Zählung der Maschen mit 1; bei Lokationen auf der Maschenhülle (auch bei Darstellung allein auf der Oberfläche) beginnt die Zählung mit 0 (auf der linken Maschenhülle der ersten Masche), 1 auf der rechten Maschenhülle der ersten Masche), usw.. Für die in degenerierten Netzen konstant gehaltenen Koordinatenrichtungen müssen KOR1 und KOR2 $= -1$ sein oder die Werte der Maschenindizes der konstant gehaltenen Koordinaten haben.

Die folgende **Tabelle 3.2.1** 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	x
3	Gsharp, n.b.	-3...+2	x		x	x	x
−3	Gsharp, c.b.	-3...-1	x		x	x	
4	AVS-UCD, n.b.	-3...-1,+1,+2	x	x	x	x	x
−4	AVS-UCD, c.b.	-3...-1	x	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 (siehe obige Anmerkungen zu IZ).
- „Null-Ko'te" bedeutet alternativ 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 (siehe obige Anmerkungen zu KNULL).

Die folgende **Tabelle 3.2.2** zeigt die Verträglichkeit der Gruppen NG für unterschiedlich belegte Netze mit den Parametern IZ und MODE.

	NG	5/15 15		5/15 15		5/15 15		7/17 17		16	19	21
	Netzbelegung	voll		Teilnetz		Löcher		beliebig		keine	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	x
±3	Gsharp	x[3]	x		x	x[3]	x	x[3,4]	x	(x)	x	x
±4	AVS-UCD	x		x		x		x				
±5	Tecplot	x	x	x	x	x	x	x[4]	x	(x)	x	x

(x): noch nicht implementiert
[3] nur für reguläre Netze und kartesische Koordinaten
[4] durch Expansion auf Gruppen 5/15 mit Löchern

3.3 Eingabebeispiele

Im folgenden werden einzelne Fälle der Steuereingabe anhand von Beispielen erläutert (<MODE> steht für das jeweils mögliche und gewählte Visualisierungssystem). Die Beispiele sind konsistent mit den im Teil 2 der VISART-Dokumentation [1b], Anhang A, aufgeführten Beispielen zum Aufbau von VISART-Dateien. Weitere Eingabebeispiele findet man in den Abschnitten 6.2.4, 6.3.4 und 6.4.4 für die einzelnen Visualisierungssysteme.

Netzgrößen (Gruppen 5/15) werden durch die Angabe ihrer Identifikation in CN eindeutig bestimmt (zu den vorgesehenen Lokationen siehe Tab. 2.3.1). Mit IKOM, KOR1 und KOR2 werden Vektorkomponenten bzw. Teilgebiete ausgewählt.
Beispiele für eine skalare Größe über dem vollen Netz, und für die Komponente einer vektoriellen Größe über einem Teilgebiet daraus, jeweils zu einem bestimmten Problemzeitpunkt:

```
<MODE> -2 0 0 0 4 4
1 1 1 1
99.9 100.1
15 'ALPLK 3' '' 0 -1 -1 -1 -1 -1 -1
```

```
<MODE> -2 0 0 0 4 4
1 1 1 1
99.9 100.1
15 'VEL   2' '' 1 -1 -1 1 4 -1 -1
```

Beispiel für eine Darstellung durch Hinzunahme der Zeitkoordinate:

```
<MODE> 2 0 0 0 4 4
0.01
1 1 1 1
100.  300.
15 'ALPLK 3' '' 0 -1 -1 -1 -1 -1 -1
```

Beispiel für die Darstellung einer Größe in einer bestimmten Masche (1,3) nur über der Zeitkoordinate:

```
<MODE> 0 0 0 0 4 4
1 0 0 1
15 'ALPLK 3' '' 0 1 1 3 3 -1 -1
```

Maschengrößen (Subgruppen 7/17) werden durch die Angabe ihrer Identifikation in CN und der Referenz auf die zugehörige Referenzgruppe 6/16 in CG eindeutig bestimmt (zu den vorgesehenen Lokationen siehe Tab. 2.3.1). Mit IKOM, KOR1 und KOR2 werden Vektorkomponenten bzw. Teilgebiete ausgewählt.
Beispiele für eine skalare Größe über dem defektiven Netz, und für die Komponente einer vektoriellen Größe über einem Teilgebiet daraus, jeweils zum letzten vorhandenen Problemzeitpunkt:

```
<MODE> -2 0 0 0 4 4
1 1 99 1
17 'ALPLK 3' 'INDEX' 0 -1 -1 -1 -1 -1 -1
```

```
<MODE> -2 0 0 0 4 4
1 1 99 1
17 'VEL   2' 'INDEX' 1 -1 -1 1 4 -1 -1
```

Punktgrößen (Gruppen 6/16) werden durch die Angabe ihrer Identifikation in CN und in CG bestimmt. Mit der Gruppe 6/16 werden nur die Koordinaten für die Visualisierung der Partikel o.ä. übergeben. Wahlweise können zugehörige Eigenschaften der Partikel o.ä. als Punktgrößen mit den Subgruppen 7/17 durch die Angabe ihrer Identifikation in CN und der Referenz auf die zugehörige Gruppe 6/16 in CG spezifiziert werden. Die Auswahl von Teilgebieten ist (noch) nicht möglich; Vektorkomponenten können nur in den Subgruppen 7/17 ausgewählt werden.
Beispiel:

```
<MODE> -2 0 0 0 4 4
1 1 99 2
16 'PARTICLE' 'PARTICLE' 0 -1 -1 -1 -1 -1 -1
17 'MASS' 'PARTICLE' 0 -1 -1 -1 -1 -1 -1
```

Integralgrößen in Gruppe 19 können auf zweierlei Weise spezifiziert werden. Zum einen kann die Position einer Größe in der betreffenden Integralgrößen-Gruppe (in KOR1(1)) direkt spezifiziert werden. Zum anderen kann diese Position automatisch bestimmt werden, falls eine Integralkonstanten-Gruppe 9 im Kopf-Paket der Postprocessor-Datei vorhanden ist, die die *Namen* der entsprechenden Werte der Integralgrößen-Gruppen 19 in den Rumpf-Paketen enthält; die Identifikation dieser Gruppe 9 muß in den ersten Zeichen mit der Identifikation der Integralgrößen-Gruppe 19 übereinstimmen. (Der Eingabeparameter CG darf nur die Zeichenkette enthalten, die der Identifikation der beschreibenden Integralkonstanten-Gruppe 9 im Kopf-Paket und der Identifikation der Integralgrößen-Gruppe 19 in den Rumpf-Paketen gemeinsam ist. Z.B. für die Identifikationen INTGRLNM (Gruppe 9) bzw. INTGRLVL (Gruppe 19) ist CG gleich INTGRL zu setzen.) Die Position der Integralgröße wird dann aus der Position ihres Namens CN in der beschreibenden Gruppe bestimmt, bei mehrfachem Auftreten des Namens außerdem zusätzlich qualifiziert durch den Eingabeparameter KOR1(1).
Beispiele für die beiden Weisen der Spezifikation einer Größe:

```
<MODE> 0 0 0 0 4 4
1 0 0 1
19 '' 'INTGRLVL' 0 5 -1 -1 -1 -1 -1
```

```
<MODE> 0 0 0 0 4 4
1 0 0 1
19 'MASS 5' 'INTGRL' 0 -1 -1 -1 -1 -1 -1
```

Zeitfunktionsgrößen (Subgruppen 21) werden durch die Angabe ihrer Identifikation in CN und, bei Netz- oder Maschengrößen, durch Angabe ihrer Maschenindizes in KOR1(1), KOR1(2) und KOR1(3) eindeutig bestimmt. Mit IKOM werden Vektorkomponenten ausgewählt.
Beispiele für eine Netzgröße und eine Integralgröße als Zeitfunktionsgrößen:

```
<MODE> 0 0 0 0 4 4
1 0 0 1
21 'ALPLK 3' '' 0 2 -1 4 -1 -1 -1
```

```
<MODE> 0 0 0 0 4 4
1 0 0 1
21 'MASL  3' '' 0 -1 -1 -1 -1 -1 -1
```

Darstellung der **Oberfläche** allein ist z.Z. nur für Maschengrößen (Subgruppen 7/17) möglich (IZ $\neq 0$; node- oder cell-based Darstellungen). Dabei können Größen, die für Lokationen auf der Netzhülle (eigentlich zum Zwecke der Überlagerung (s.u.) mit den zugehörigen Größen im Netzinnern) gegeben sind, auch allein (ohne letztere) dargestellt werden (siehe Tab. 2.3.2). Dazu ist der Eingabeparameter IV $= \pm 1$.
Beispiel für eine skalare Größe über dem defektiven Netz (Oberfläche auf den Maschenhüllen zwischen $j = 1$ und $j = 2$):

```
<MODE> -2 0 0 -1 4 4
'+2' 'S NORMV'
1 1 99 1
17 'SALPLK3' 'S INDEX' 0 -1 -1 1 1 -1 -1
```

Überlagerung ist für Netzgrößen (Gruppen 5/15)—jedoch nicht bei Teilnetzen—und Maschengrößen (Subgruppen 7/17) möglich (IZ $\neq 0$; node-based Darstellungen). Dabei können Größen, die in den Maschenmitten im Innern des Netzes lokalisiert sind, mit den zugehörigen Größen auf der Netzhülle kombiniert werden (siehe Tab. 2.3.3). Dazu ist der Eingabeparameter IV ≥ 2 oder ≤ -2. Eine Gruppe mit einer Größe auf der Netzhülle wird in der Steuereingabe unmittelbar hinter der Eingabezeile für die Gruppe mit der zugehörigen Größe im Netzinnern aufgeführt. Zur Verifizierung der Zusammengehörigkeit und zur Erzeugung der Labels (siehe 4.2) dürfen sich die Identifikationen der zusammengehörenden Größen nur in Prefixen oder Suffixen unterscheiden (siehe Eingabeparameter CC). Die Eingabeparameter IKOM, KOR1(n), und KOR2(n), für $n = 1, 2, 3$, müssen bei den zusammengehörenden Größen gleich sein.
Beispiel für eine skalare Größe über dem defektiven Netz:

```
<MODE> -2 0 0 -2 4 4
'' 'S ' 'S NORMV'
1 1 99 2
17 'ALPLK 3' 'INDEX' 0 -1 -1 -1 -1 -1 -1
17 'SALPLK3' 'S INDEX' 0 -1 -1 -1 -1 -1 -1
```

Die folgenden **Tabellen 3.3.1** und **3.3.2** fassen noch einmal zusammen, wie die Zeilen 7' der VISAPTER-Steuereingabe für eine Überlagerung, je nach der Aufteilung der Gruppen auf Kopf- und Rumpf-Pakete, angegeben werden (sowie, welche Werte von IRAND jeweils dazu passen). Im Falle der Subgruppen 7/17 wird auch noch danach unterschieden, ob die zugehörigen Referenzgruppen 16 in den Rumpf-Paketen mit Daten belegt oder leer sind, weil eine komplementäre Referenzgruppe 6 vorhanden ist (siehe Tab. B.3 in [1b]).

Dateiaufbau	IRAND	Zeilen 7' (NG, CN, ...) für Größen der Gruppen 5/15
15 *Ident-i* 15 *Ident-s* ...	0, 1	15 *'Ident-i'* ... 15 *'Ident-s'* ... ...
5 *Ident-s* 15 *Ident-i* ...	2, 3	15 *'Ident-i'* ... 5 *'Ident-s'* ... ...

Dateiaufbau	IRAND	Zeilen 7' (NG, CN, CG, ...) für Größen der Subgruppen 7/17
16 *Ref-i* 17 *Ident-i* 16 *Ref-s* 17 *Ident-s* ...	0, 1	17 *'Ident-i'* *'Ref-i'* ... 17 *'Ident-s'* *'Ref-s'* ... ...
6 *Ref-i* 6 *Ref-s* 16 *Ref-i0* 17 *Ident-i* 16 *Ref-s0* 17 *Ident-s* ...	2, 3	17 *'Ident-i'* *'Ref-i'* ... 17 *'Ident-s'* *'Ref-s'* ... ...
6 *Ref-i* 6 *Ref-s* 7 *Ident-s* 16 *Ref-i0* 17 *Ident-i* ...	2, 3	17 *'Ident-i'* *'Ref-i'* ... 7 *'Ident-s'* *'Ref-s'* ... ...

Anmerkungen:

Ident-i steht für die Identifikation einer Gruppe 15 oder einer Subgruppe 17 mit Größen im Netzinnern;
Ident-s steht für die Identifikation einer Gruppe 5/15 oder einer Subgruppe 7/17 mit Größen auf der Netzhülle;
Ref-i steht für die Identifikation einer Referenzgruppe 6/16 für Subgruppen 7/17 mit Größen im Netzinnern;
Ref-s steht für die Identifikation einer Refernzgruppe 6/16 für Subgruppen 7/17 mit Größen auf der Netzhülle;
Ref-i0 und *Ref-s0* stehen für die entsprechenden leeren Gruppen.

3.4 Eingabe über die Benutzeroberfläche

Wenn das Programm VISAPTER über die grafische Benutzeroberfläche [1a] aufgerufen wird, wird die Steuereingabe durch das Tcl/Tk-Skript der Benutzeroberfläche aus den Einstellungen der „Widgets" im VISAPTER-Fenster und in den zugehörigen Unterfenstern aufgebaut und nach Betätigung der Tasten „Ausführen" oder „n Eingaben" auf die Datei `VISART_IN` im jeweiligen Arbeitsverzeichnis geschrieben. Diese Datei wird dem Programm beim Aufruf als Eingabedatei übergeben.

Die Widgets (das sind Tasten, Wähltasten, Schalttasten, Menütasten, Eingabekästchen, Anzeigefelder) sind meist einem bestimmten Eingabeparameter zugeordnet und setzen dessen Wert nach der Stellung der Taste(n) oder auf den im Kästchen oder Anzeigefeld eingegebenen Wert. Einige Tasten öffnen nach Betätigung ein Unterfenster für die Einstellung weiterer Gruppen von Parametern. Die Funktion der Widgets im VISAPTER-Fenster ist durch in die VISART-Benutzeroberfläche eingebaute Hilfetexte dokumentiert. In diesen Hilfen werden auch die Zuordnung der Widgets zu den Eingabeparameter und die den Tastenstellungen entsprechenden Werte angegeben.

Bild 2 zeigt das VISAPTER-Fenster, das für den Aufruf von VISAPTER mit dem Eingabeparameter `MODE` $\neq 0$ in Frage kommt. Alle Widgets unterhalb der Anzeigefelder für die Dateien, mit Ausnahme der Schalttaste „ASCII" (siehe 5.2), dienen zur Eingabe von Steuerparametern. Mit der Wähltastenleiste für das Visualisierungssystem wird der Betrag des Parameters `MODE` gesetzt, mit der Schalttaste „cell-based" dessen Vorzeichen. Mit dem Wähltastenfeld für die Darstellungskoordinaten wird der Parameter `IZ` gemäß folgender Tabelle spezifiziert:

	x, y, z	x, y	x	t	x, t	x, y, t
IZ	-3	-2	-1	0	1	2

Mit der Wähltastenleiste „keine...", „Umrechnung", „Abwicklung", „Spiegelung" wird der Parameter `IKRUM` spezifiziert. Im Kästchen „Zeitfaktor" wird der Parameter `TX` eingesetzt, in den Kästchen der nachfolgenden Zeile die „Anzahl" `NP` der Rumpf-Pakete, die Anzahl `NT` der spezifizierten Problemzeit-„Intervalle" und die entsprechenden „Intervallgrenzen" `T1`, `T2`, schließlich in einem Kästchen die „Anzahl" `NN` der spezifizierten Größen. (Der Parameter `NF` wird implizit über die Spezifizierung der VISART-Datei(en) im linken oberen Anzeigefeld gesetzt.) In den Kästchen des letzten Feldes werden die Parameter für die `|NN|` Größen eingesetzt, und zwar für jede Größe eine Zeile mit `NG`, `CN`, `CG`, `IKOM`, `KOR1(1)`, `KOR2(1)`, `KOR1(2)`, `KOR2(2)`, `KOR1(3)`, `KOR2(3)`, falls `NN` > 0 ist, oder `CI`, falls `NN` < 0 ist.

Weitere Eingabeparameter werden in Unterfenstern spezifiziert. Bild 2.a zeigt das mit der Taste „Null-Ko'te" gekoppelte Unterfenster zur Eingabe der Parameter `KNULL` > 0 und `FORMT`. Bild 2.b zeigt das mit der Taste „1D/2D in 3D" gekoppelte Unterfenster zur Eingabe des Parameters `KNULL` < 0. Bild 2.c zeigt das mit der Taste „Oberflaeche" gekoppelte Unterfenster zur Eingabe der Parameter `IV` und `CC` bei Alleindarstellung von Oberflächen. Bild 2.d zeigt das mit der Taste „Ueberlagerung" gekoppelte Unterfenster zur Eingabe der Parameter `IV` und `CC` bei Überlagerung von Randwerten. Da im VISAPTER-Fenster nur Platz für die Spezifizierung von maximal 4 Größen ist, kann durch Doppelklicken auf das Kästchen für die Anzahl der Größen ein Unterfenster für die Spezifizierung von bis zu 20 zu übernehmender bzw. nicht zu übernehmender Größen eröffnet werden.

Identifikationen (für `CC`, `CNORM`, `FORMT`, `CN`, `CG`, `CI`) werden in die Kästchen der Benutzeroberfläche ohne die bei der listengesteuerten Eingabe unter Fortran benötigten Apostrophe eingesetzt. In den Identifikationen stehende Leerzeichen werden deshalb als Unterstriche eingegeben. Sind Unterstriche Teil der Identifikation, so sind sie bei der Eingabe durch einen vorgesetzten Backslash zu „quoten".

Für nicht ausgefüllte Kästchen werden den Parametern Default-Werte zugewiesen, falls dies möglich ist. Die folgenden Eingabeparameter können im VISAPTER-Fenster nicht spezifiziert

werden, sondern werden im zugehörigen Tcl/Tk-Skript der Benutzeroberfläche auf feste Werte gesetzt:

IRAND = 4
IRAKD = 4

(Infolgedessen können beim Aufruf von VISAPTER über die Benutzeroberfläche zur Zeit nur zeitlich konstante Netze verarbeitet werden.)

Einige weitere Eingabeparameter sind wegen des begrenzten Platzes in den Fenstern in der Größe eingeschränkt:

$|IV| \leq 4$
$|NT| \leq 3$
$|NN| \leq 20$

Bild 3 zeigt das VISAPTER(0)-Fenster zur Werteinspektion, das eine vereinfachte Version des VISAPTER-Fensters für den Eingabeparameter MODE = 0 ist. Mit dem Wähltastenfeld für die Darstellungskoordinaten wird hier der Parameter IZ gemäß folgender Tabelle spezifiziert:

	Geometrie	Zeit	Geom+Zeit
IZ	-9	0	9

Außer den beim VISAPTER-Fenster angeführten werden die folgenden Eingabeparameter im Tcl/Tk-Skript für das VISAPTER(0)-Fenster auf feste Werte gesetzt:

MODE = 0
IV = 0
KNULL = 0
TX = 1.0

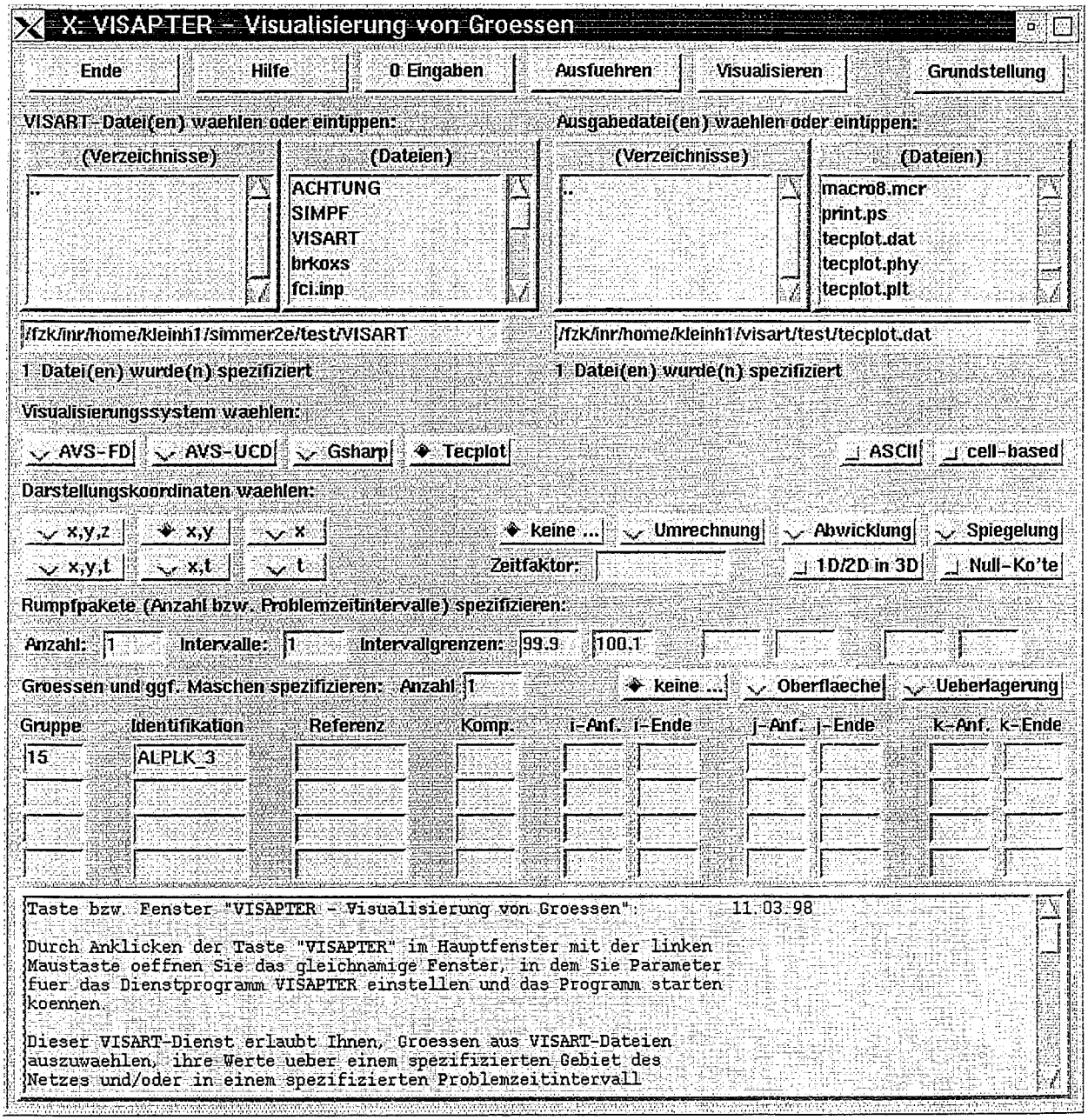


Bild 2: VISAPTER-Fenster

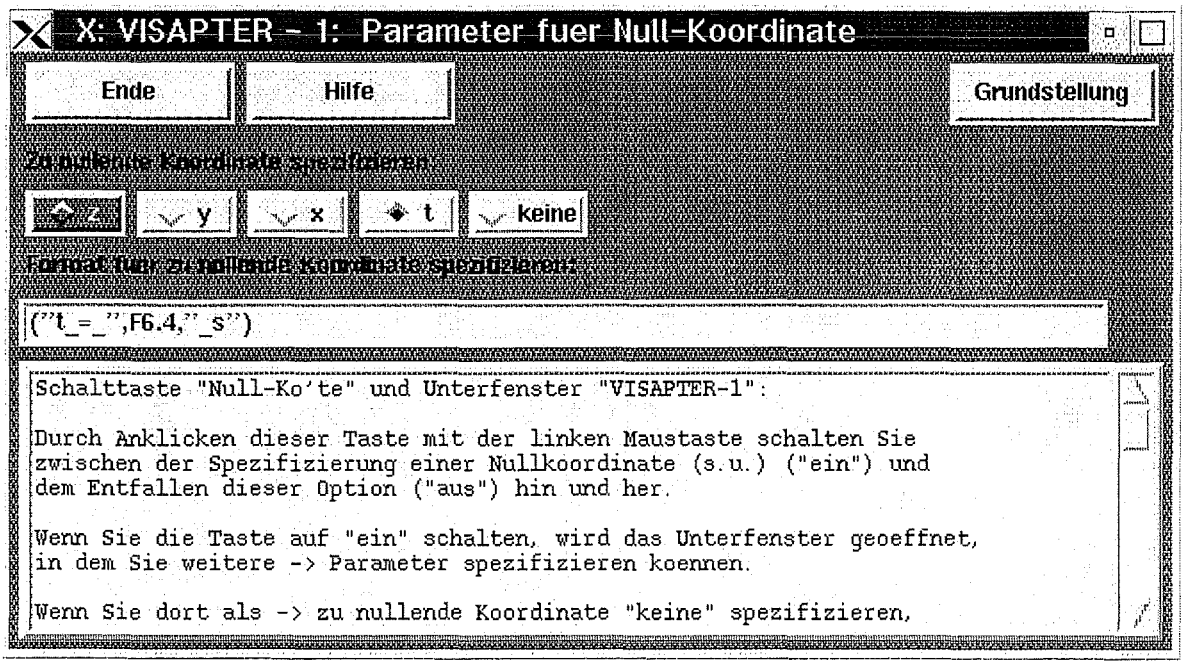


Bild 2.a: VISAPTER-Unterfenster für Null-Koordinate

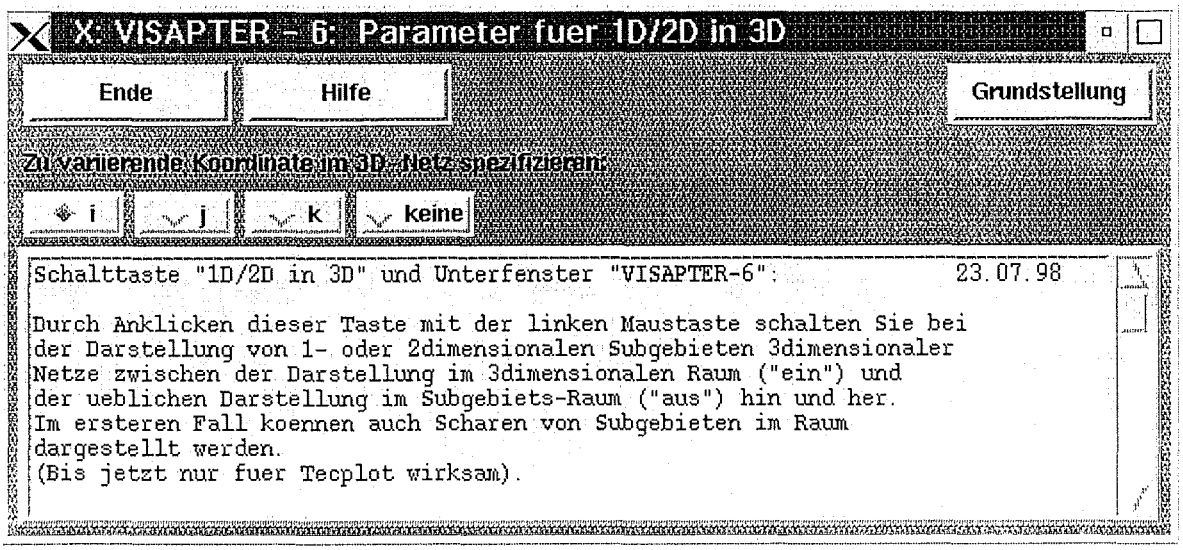


Bild 2.b: VISAPTER-Unterfenster für 1D/2D in 3D

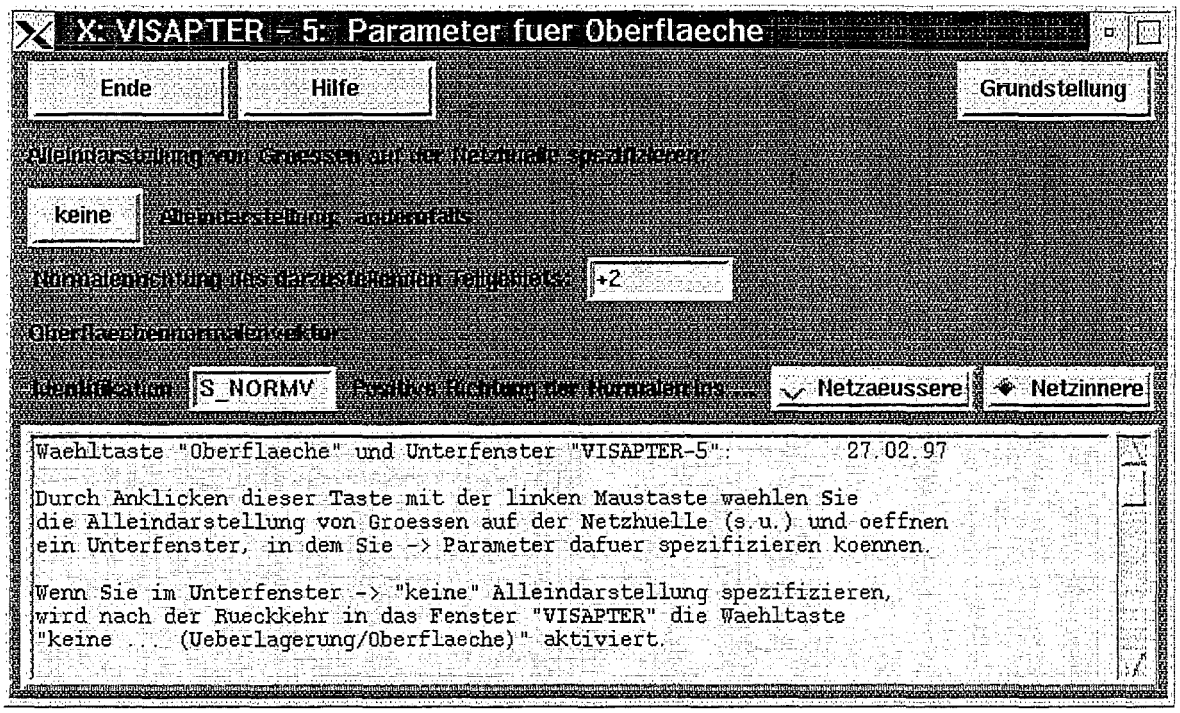


Bild 2.c: VISAPTER-Unterfenster für Oberfläche

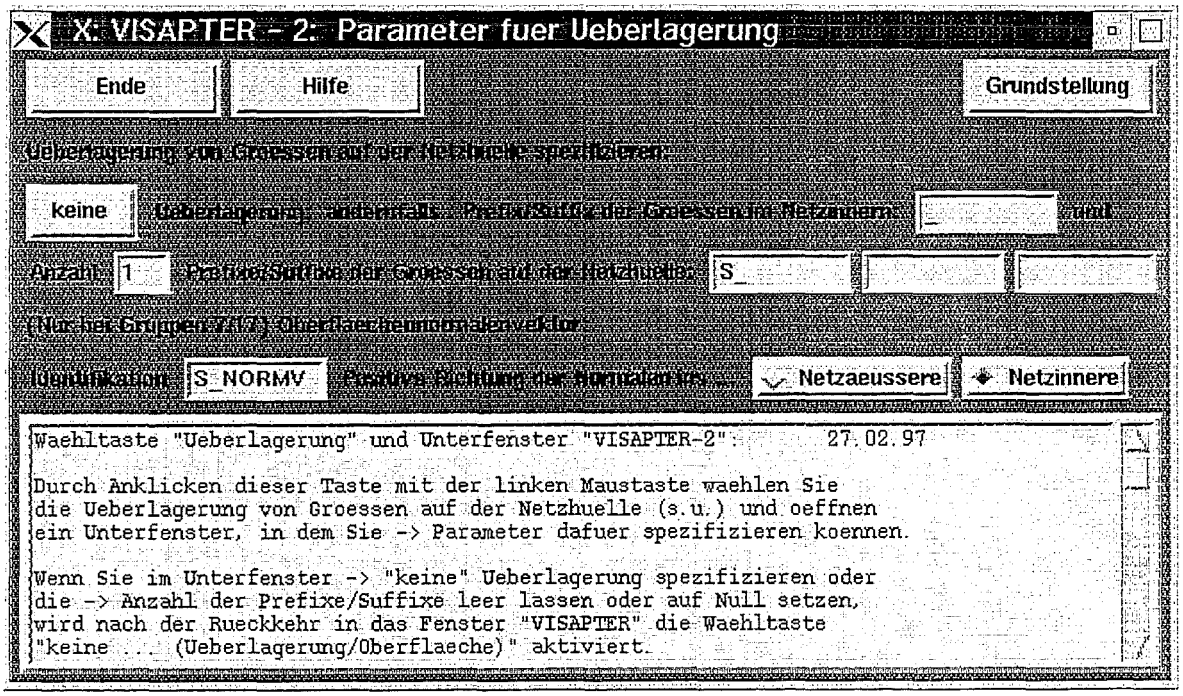


Bild 2.d: VISAPTER-Unterfenster für Überlagerung

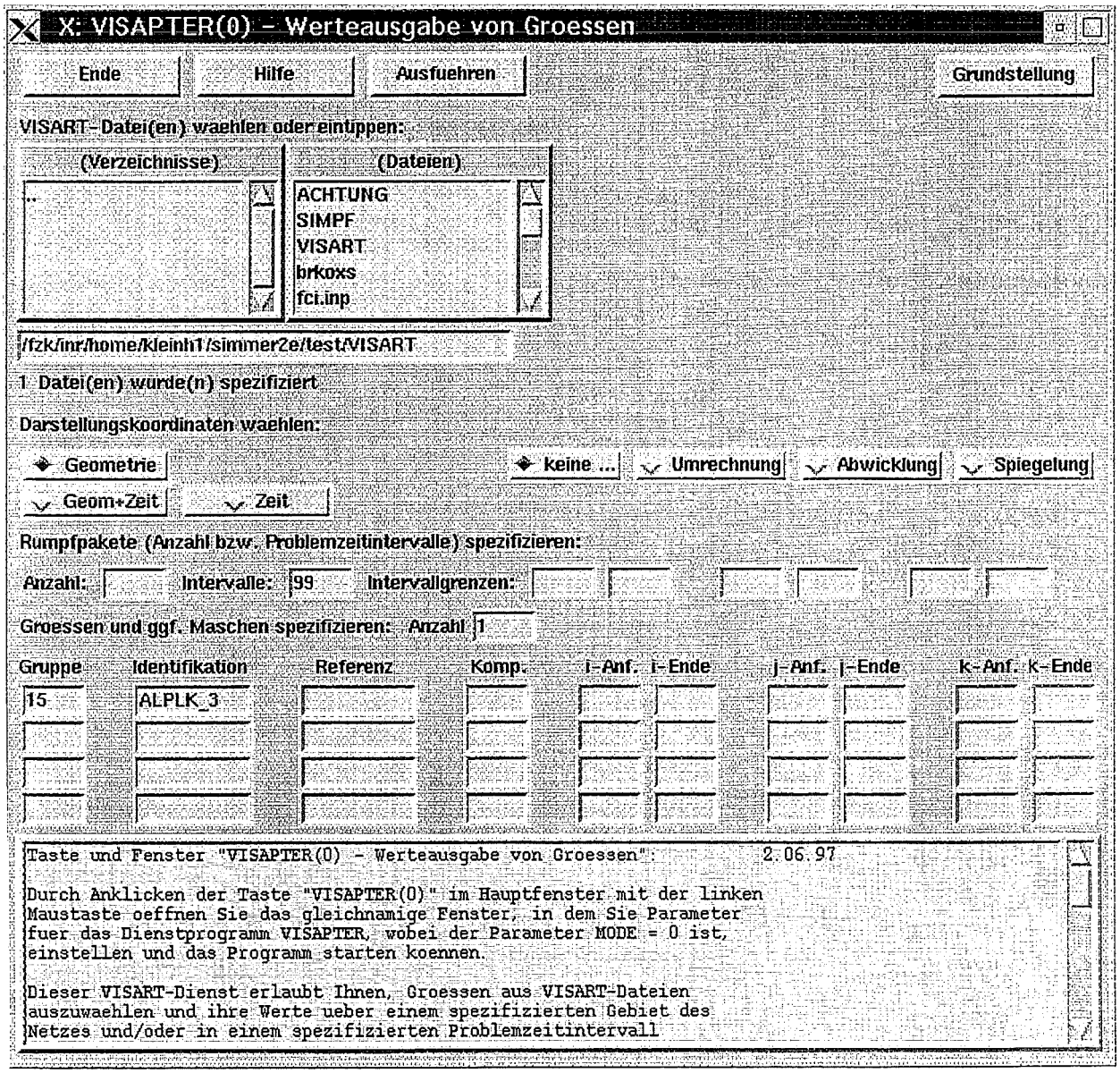


Bild 3: VISAPTER(0)-Fenster

4. Ein- und Ausgabedateien des Programms

In diesem Kapitel werden die VISART-Eingabedatei(en) des Programms VISAPTER und seine Ausgabedateien für die Visualisierungssysteme beschrieben.

4.1 VISART-Dateistruktur und -inhalt

Im VISART-Standard ist nur die Form, nicht aber der Inhalt der Postprocessor-Dateien festgelegt. Allerdings werden von VISAPTER einige wenige Einschränkungen der Struktur der Dateien und einige Konventionen vorausgesetzt, die im folgenden aufgeführt werden.

Bezüglich Anzahl und Format der Dateien gilt:

- Die von VISAPTER erwarteten VISART-Dateien können aus einer oder mehreren Anwendungen eines Codes stammen, und jeweils aus einem Erstlauf und beliebig vielen Restart-Läufen einer Anwendung bestehen. Solche Fortsetzungsdateien werden hier als „Files“ *einer* VISART-„Datei“ bezeichnet. VISAPTER akzeptiert eine oder mehrere „Dateien“ im obigen Sinne, mit jeweils einem oder mehreren „Files“, als Eingabe (siehe 3.1).
- Die von VISAPTER 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 VISAPTER unter dem gleichen Betriebssystem und dem gleichen Processor-Typ läuft, wie diejenigen, mit denen die Dateien geschrieben wurden.
- Da VISAPTER in Fortran geschrieben ist, müssen unformatierte VISART-Dateien, die z.B. mit C-Programmen erstellt werden, auch die interne Darstellung der Fortran-Records nachbilden (mit einem 4-Byte-Wort am Anfang und Ende jedes Records, in dem die Anzahl der Bytes des Records steht), wenn sie von VISAPTER gelesen werden sollen.

Bezüglich der Struktur gilt:

- VISAPTER erwartet, daß alle Rumpf-Pakete einer VISART-Datei gleichartig aufgebaut sind, d.h. aus der gleichen Anzahl und Anordnung gleicher Gruppen bestehen. Dies gilt auch über alle Fortsetzungs-Files (aus Restart-Läufen eines Codes) *einer* VISART-Datei hinweg. Bei mehreren VISART-Dateien (im obigen Sinne) als Eingabe brauchen die Dateien untereinander nicht gleichartig aufgebaut sein.
- VISAPTER kann Größen der Gruppen/Subgruppen 5/15, 6/16, 7/17, 9/19, 20 und 21 aus VISART-Dateien verarbeiten, soweit dies vom Visualisierungssystem her möglich ist (siehe Tab. 3.2.2). Größen der Gruppen/Subgruppen 51/151, 61/161 und 71/171 werden dagegen bisher noch nicht akzeptiert.
- In einem Kopf- oder Rumpf-Paket können mehrere Referenzgruppen 6/16, mit jeweils zugehörigen Subgruppen 7/17, auftreten. In jedem Paket ist aber nur *eine* Referenzgruppe 20, mit zugehörigen Subgruppen 21, erlaubt.
- Bei der Angabe von Größen der Gruppen 5/15 über Teilnetzen [1a, 1b] müssen die Identifikationen einer Größe in allen Gruppen, die jeweils ein Teilnetz überdecken, die gleichen sein. Die Reihenfolge der Teilnetze, gegeben durch die Reihenfolge der Gruppen, muß für alle Größen die gleiche sein. Die Teilnetze können, aber müssen sich nicht, überlappen.

Bezüglich der Konventionen gilt:

- VISAPTER erwartet die Geometrie-Gruppe 4 unter der Identifikation `GEOMETRIE`.
- Bei irregulären Netzen müssen die Gruppen 5/15 oder 7/17 mit den Netzkoordinaten für die benötigten Lokationen Identifikationen mit dem Namensbestandteil `..COORD..` tragen.
- Bei „schwach defektiven" Netzen müssen die Gruppen 5/15 mit der Angabe der nicht belegten Maschen („Löcher") des Netzes Identifikationen mit dem Namensbestandteil `..DEFCT..` tragen. In diesen Gruppen (Lokation 0, Typ Integer) stehen für die belegten Maschen Nullen, für die Löcher Einsen.
- Bei defektiven Netzen müssen die Referenzgruppen 6/16 mit den Maschenindexpaaren/tripel der belegten Maschen oder Maschenhüllen Identifikationen mit dem Namensbestandteil `..INDEX..` tragen.
- Zur Beschreibung von Größen der Lokationen -11, -22 und -44 auf der Netzhülle eines defektiven Netzes wird für jede dieser Lokationen eine eigene Referenzgruppe 6/16 benötigt.
- Zur Beschreibung von Größen der Lokationen -33 oder -77 auf der Netzhülle eines defektiven Netzes wird eine eigene Referenzgruppe 6/16 benötigt. Dieser muß als erste eine Subgruppe 7/17 mit dem Oberflächennormalenvektor der Netzhülle folgen, die eine Identifikation mit dem Namensbestandteil `..NORMV..` tragen muß.
- Wenn Werte von Größen auf der Netzhülle solchen im Netzinneren überlagert werden sollen (siehe 2.3), dürfen sich die Identifikationen zusammengehöriger Größen nur durch Prefixe oder Suffixe unterscheiden. Die Prefixe bzw. Suffixe sind für alle Größen im Netzinneren und für alle Größen der gleichen Lokation auf der Netzhülle die gleichen (bei defektiven Netzen gilt dies auch für die Identifikation der zugeordneten Referenzgruppe 6/16 und der Subgruppe 7/17 mit dem Oberflächennormalenvektor).

Bezüglich der Verteilung auf Kopf-Paket und Rumpf-Pakete der VISART-Datei gilt:

- Für VISAPTER können die Gruppen mit den Angaben zur Netzgeometrie (Maschenspezifizierung, -belegung, -koordinaten) in der VISART-Datei an verschiedener Stelle stehen: 5 bzw. 6/7 im Kopf-Paket, 15 bzw. 16/17 im ersten Rumpf-Paket oder 15 bzw. 16/17 in jedem Rumpf-Paket, was sich danach richtet, ob die Geometrie und die Größen auf der Netzhülle über der Problemzeit konstant bleiben oder sich ändern. Im einzelnen gilt:
- Die Referenzgruppen 6/16 mit der Angabe der belegten Maschen oder Maschenhüllen bei defektiven Netzen (sowie die Subgruppe 7/17 mit dem Oberflächennormalenvektor) können sowohl im Kopf-Paket, als auch in den Rumpf-Paketen stehen. Stehen sie als Gruppen 6 im Kopf-Paket, so dürfen die (syntaktisch erforderlichen) entsprechenden Gruppen 16 in den Rumpf-Paketen leer sein (siehe Tab. 3.3.2, sowie B.3 in [1b]).
- Die Gruppen mit den Netzkoordinaten (Gruppen 5/15 oder Subgruppen 7/17), sowie mit der Angabe der nicht belegten Maschen bei „schwach defektiven" Netzen (Gruppen 5/15) können sowohl im Kopf-Paket als auch in den Rumpf-Paketen stehen (siehe Tab. B.2 in [1b]).

4.2 Ausgabedateien

Die Ausgabedateien von VISAPTER dienen als Eingabedateien für die Visualisierungssysteme. Sie sind formatiert, d.h. in ASCII, geschrieben, außer für Tecplot, wo auch eine unformatierte, d.h. binäre, Form möglich ist (siehe 5.1 und 5.2). In der Regel wird nur *eine* Datei für das Visualisierungssystem benötigt, zwei Dateien lediglich für AVS-FD (sowie ggf. eine weitere für den Koordinaten-File—siehe letzter Absatz).

Die Ausgabedateien enthalten für alle Systeme die Koordinaten, Werte und Labels der aus der VISART-Datei ausgewählten Größen über dem spezifizierten Teilgebiet und/oder den spezifizierten Problemzeiten. Deren Anordnung und weitere Details unterscheiden sich bei den einzelnen Systemen (siehe 6.2.2, 6.3.2, 6.4.2).

In der Beschreibung des Programms und in den Ausgabedateien wird mit „Komponente" schlechthin eine skalare Größe und jede Vektorkomponente einer vektoriellen Größe bezeichnet. Überlagerte Größen werden zu einer Komponente zusammengefaßt. Den Komponenten werden die auszugebenden Größen, falls sie mit `NN` > 0 explizit in der Steuereingabe spezifiziert sind, in der spezifizierten Reihenfolge zugeordnet; andernfalls in der Reihenfolge der Größen in der Postprocessor-Datei. Diese Reihenfolge der Größen gilt auch auf den Ausgabedateien für alle Visualisierungssysteme. Bei vektoriellen Größen erscheinen jeweils die Vektorkomponenten der Größen nacheinander (bei `IKOM` < 0 erscheinen nur die ersten `|IKOM|` der im Netz bzw. Teilgebiet liegenden Vektorkomponenten).

Zur Beschriftung bei der Visualisierung erzeugt VISAPTER für jede Komponente Labels aus maximal 16 Zeichen. Diese bestehen aus der Identifikation der Größe (bei Überlagerung ohne Prefix bzw. Suffix—siehe 3.3), ggf. ergänzt um die Vektorkomponentennummer `IKOM` (nach einem At-Zeichen `@`) und/oder um die Maschenindizes festgehaltener Koordinatenrichtungen (in Klammern, mit Sternen `*` für die freien Maschenindizes) bei Darstellungen über Subgebieten bzw. um die Position (in Klammern) einer Größe in einer Gruppe 19.

Neben den Eingabedateien für die Visualisierungssysteme kann VISAPTER bei AVS-FD und AVS-UCD noch eine weitere ASCII-Datei erstellen, falls der Eingabeparameter `KNULL` > 0 ist, nämlich den "Koordinaten-File" (siehe 5.2), mit den Koordinatenwerten der durch `KNULL` bezeichneten Koordinate im Format `FORMT` (siehe 4.2).

5. Technik des Programms

In diesem Kapitel werden Aufruf, Aufbau und Charakteristiken des Programms VISAPTER beschrieben.

5.1 Programmoptionen

Das Programm VISAPTER ist in Fortran geschrieben und damit in jedem Betriebssystem implementierbar. Es läuft zur Zeit unter UNIX auf IBM-RS/6000-Workstations (sowie unter MVS auf IBM-Großrechnern, wofür es ursprünglich entworfen wurde).

Das Programm verwendet im wesentlichen Fortran-77; aus Fortran-90 wurden für eine dynamische Speicherverwaltung einige Element im Zusammenhang mit ALLOCATE-Befehlen und das Module-Konzept übernommen.

Die Versionsführung geschieht mit dem Werkzeug HISTORIAN [2]. Die aktuelle HISTORIAN-Source von VISAPTER (z.Z. Release D.4) steht unter DCE/DFS in der Datei

```
/fzk/inr/@sys/VISART/sources90/visapter90
```

Aus ihr wird in einem HISTORIAN-Lauf zunächst eine compilierbare Fortran-Source, unter Berücksichtigung der spezifizierten Optionen, erstellt. Die folgenden Optionen können kombiniert werden:

- `UNIX`: Falls diese Option „on" ist, wird die Fortran-Source für UNIX erzeugt (andernfalls für MVS).
- `ENGLISH`: Falls diese Option „off" ist, wird eine Fortran Source mit deutschsprachigen Nachrichten und Fehlermeldungen im Ausführungsprotokoll erzeugt; andernfalls mit englischsprachigen Texten.
- `DBLPP`: Zum Lesen von unformatierten VISART-Dateien mit einfacher Wortlänge der `REAL`-Variablen und von formatierten VISART-Dateien [1b] muß diese Option „off" sein (das Programm erkennt aus der ersten Beschreibungsgruppe der VISART-Datei, ob es sich um eine unformatierte oder um eine formatierte Datei handelt, und stellt sich demgemäß darauf ein). Zum Lesen und Verarbeiten von unformatierten VISART-Dateien mit doppelter Wortlänge der `REAL`-Variablen muß diese Option „on" sein.
- `BINARY`: Diese Option betrifft z.Z. nur das Format der Ausgabedatei für das Visualisierungssystem Tecplot. Wenn die Option „off" ist, wird die Datei im ASCII-Format erstellt und bei der Verarbeitung in Tecplot dort in ein binäres Format umgewandelt. Andernfalls wird die Datei bereits von VISAPTER als binäre Datei erstellt, was das Einlesen in Tecplot beträchtlich beschleunigt.

Zum automatischen Erzeugen der Fortran-Source mit unterschiedlichen Kombinationen von Optionen stehen unter DCE/DFS Shell-Scripts unter den Namen

```
/fzk/inr/@sys/VISART/sources90/visapter??.s
```

zur Verfügung, wo `??` die Optionenkombination bezeichnet (siehe 5.2). (Die Shell-Scripts setzen auch Datum und Uhrzeit der Fortran-Source-Erstellung in die dafür vorgesehenen `DATA`-Statements in VISAPTER ein, deren Inhalt bei der Ausführung des Programms ins Protokoll gedruckt wird.)

5.2 Programmaufruf und -argumente

Ausführbare Binärprogramme von VISAPTER stehen unter DCE/DFS im Verzeichnis

```
/fzk/inr/@sys/VISART/bin
```

und zwar als Datei

- `visapter` für die HISTORIAN-Option `UNIX`;
- `visapter-e` für die HISTORIAN-Optionen `UNIX, ENGLISH`;
- `visapter-b` für die HISTORIAN-Optionen `UNIX, BINARY`;
- `visapter-v` für die HISTORIAN-Optionen `UNIX, ENGLISH, BINARY`.

Beim Aufruf von VISAPTER aus der VISART-Benutzeroberfläche werden bei Sprachauswahl „deutsch“ (über die Menütaste „Sprache“ im Hauptfenster der Benutzeroberfläche) die Binärprogramme ohne die Option `ENGLISH` ausgeführt, bei Sprachauswahl „english“ die Binärprogramme mit der Option `ENGLISH`. Bei gedrückter Taste „ASCII“ im VISAPTER-Fenster werden die Binärprogramme ohne die Option `BINARY` ausgeführt, bei nicht gedrückter Taste die Binärprogramme mit der Option `BINARY`.

Das Programm VISAPTER erwartet die Steuereingabe auf seiner Standardeingabe (Unit 5) und schreibt das Ausführungsprotokoll auf seine Standardausgabe (Unit 6). Das Ausführungsprotokoll besteht aus der Wiedergabe der Steuereingabe, dem Ausdruck der Kennsätze der ausgewählten Gruppen, und eventuellen Fehlermeldungen (und ggf. der Ausgabe der Zahlenwerte der Größen, falls diese anstelle der Visualisierung gewählt wurde).

Beim Aufruf von VISAPTER über die grafische Benutzeroberfläche wird die Steuereingabe der von den Tcl/Tk-Skripts der Benutzeroberfläche aufgebauten Datei `VISART_IN` entnommen (siehe 3.4). Die Protokollausgabe wird auf die Datei `VISART_OUT` im jeweiligen Verzeichnis gelegt und im Textfeld des VISAPTER-Fensters angezeigt. Eventuelle Fehlerausgabe wird auf die Datei `VISART_ERR` im jeweiligen Verzeichnis gelegt und in eigenen Dialogfenstern angezeigt.

Neben der Steuereingabe erwartet VISAPTER eine oder mehrere VISART-Datei(en) als Eingabe, jeweils aus einem oder mehreren Files bestehend (siehe 4.1), und erstellt eine oder mehrere Dateien für die Visualisierungssysteme (siehe 4.2). Die Namen der laut Steuereingabe benötigten Ein- und Ausgabedateien werden als Argumente beim Programmaufruf angegeben:

```
visapter [ 10/1 [ 10/2 ... ]  [ 20 [ 21 [ 22 ] ] ] ]
```

wo die Unit-Nummern für die Datei-/File-Namen laut nachfolgender Tabelle 5.2 stehen. Beim Aufruf von VISAPTER über die grafische Benutzeroberfläche werden die im VISAPTER-Fenster eingestellten VISART-Datei(en) bzw. -Fortsetzungsfiles und die eingestellte(n) Ausgabedatei(en) als Argumente übergeben.

Falls keine Datei-/File-Namen als Argumente angegeben werden, verwendet VISAPTER die in der folgenden **Tabelle 5.2** stehenden:

MODE	Unit	Name	Status	Format	Bedeutung
	10/1	VISART1	old	form./unform.	1. Datei, 1. File
	10/2	VISART2	old	form./unform.	1. Datei, 2. File
	...	...	...	...	...
	10/n	VISARTn	old	form./unform.	1. Datei, n. File
	11/1	VISARTn+1	old	form./unform.	2. Datei, 1. File
	11/2	VISARTn+2	old	form./unform.	2. Datei, 2. File
	...	...	...	...	...
	11/m	VISARTn+m	old	form./unform.	2. Datei, m. File
	...	...	...	...	...
	20	DATA	unknown	formatted	AVS-FD Data File
1	21	FLD	unknown	formatted	AVS-FD Description File
	22	KOR	unknown	formatted	Koordinaten-File
±3	20	DAT	unknown	formatted	Gsharp File
±4	20	INP	unknown	formatted	AVS-UCD File
	22	KOR	unknown	formatted	Koordinaten-File
±5	20	DAT	unknown	form./unform.	Tecplot Data File

5.3 Programmaufbau

Das Hauptprogramm von VISAPTER steuert den Programmablauf in folgenden Phasen:

(Initialisierung:)

- Initialisieren und Einlesen der Zeilen 1 bis 4 der Steuereingabe (Subroutine EING1).

(1) Einlesen der Zeilen 5 bis 7 der Steuereingabe für die jeweilige VISART-Datei (Subroutine EING2); falls Eingabeende: weiter bei (9).

(2) Öffnen des anstehenden Files der VISART-Datei; Lesen der Beschreibungsgruppen; Auswerten der Geometrie-Gruppe (ggf. mit Subroutine CGEOM); Allokieren der dimensionsabhängigen Felder.

(Für das Kopf-Paket des VISART-Files:)

- Lesen des Kennsatzes der nächsten Gruppe vom VISART-File.

  Falls Gruppe 5: Aufruf von Subroutine MESH. (Dort Aufruf von Subroutine CMESH, falls die Gruppe Koordinaten definiert; von Subroutine CDEFCT, falls die Gruppe Fehlstellen definiert.) Falls die Gruppe übernommen werden soll: Einlesen der Daten, ggf. Ausschneiden des Teilgebiets, Umrechnen und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei.

  Falls Gruppe 6: Aufruf von Subroutine INDEX. Dort, falls die Gruppe Partikelkoordinaten definiert und übernommen werden soll: Einlesen der Daten, ggf. Ausschneiden des Teilgebiets, Umrechnen und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei. Falls die Gruppe Referenzgruppe für Subgruppe(n) 7 der Eingabe ist: Einlesen der Daten und Abspeichern als Indexvektor auf eine Hilfsdatei.

  Falls Subgruppe 7: Aufruf von Subroutine CELL. (Dort ggf. Speichern als Oberflächennormalenvektor oder Aufruf von Subroutine CCELL, falls die Subgruppe Koordinaten definiert.) Falls die Subgruppe übernommen werden soll: Einlesen der Daten, ggf. Ausschneiden des Teilgebiets, Umrechnen und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei.

  Falls Gruppe 9 und IZ = 0:
  Falls die Gruppe Namen enthält: Aufruf von Subroutine INTGRO. Dort Umrechnung der Identifikationen zu Gruppen 9/19 der Steuereingabe in relative Adressen.
  Andernfalls: Aufruf von Subroutine INTGR. Dort, falls eine Größe der Gruppe übernommen werden soll: Einlesen der Daten und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei.

(Für jedes Rumpf-Paket des VISART-Files:)

- Lesen der Zyklus-Gruppe des nächsten Rumpf-Pakets vom VISART-File.
  Falls der Problemzyklus/zeitpunkt nicht übernommen werden soll: Überlesen des gesamten Rumpf-Pakets.
  Falls der Problemzyklus/zeitpunkt übernommen werden soll:

- Lesen des Kennsatzes der nächsten Gruppe vom VISART-File.

  Falls Gruppe 15: Aufruf von Subroutine MESH. (Dort Aufruf von Subroutine CMESH, falls die Gruppe Koordinaten definiert; von Subroutine CDEFCT, falls die Gruppe Fehlstellen definiert.) Falls die Gruppe übernommen werden soll: Einlesen der Daten, ggf. Ausschneiden des Teilgebiets, Umrechnen und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei.

  Falls Gruppe 16: Aufruf von Subroutine INDEX. Dort, falls die Gruppe Partikelkoordinaten definiert und übernommen werden soll: Einlesen der Daten, ggf. Ausschneiden des

Teilgebiets, Umrechnen und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei. Falls die Gruppe Referenzgruppe für Subgruppe(n) 17 der Eingabe ist: Einlesen der Daten und Abspeichern als Indexvektor auf eine Hilfsdatei.

Falls Subgruppe 17: Aufruf von Subroutine `CELL`. (Dort ggf. Speichern als Oberflächennormalenvektor oder Aufruf von Subroutine `CCELL`, falls die Subgruppe Koordinaten definiert.) Falls die Subgruppe übernommen werden soll: Einlesen der Daten, ggf. Ausschneiden des Teilgebiets, Umrechnen und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei. Letzteres geschieht in `CELL` für die belegten Maschen des Netzes, falls AVS-UCD als Visualisierungssystem spezifiziert wurde, aber durch Verzweigung nach `MESH` (und anschließende Rückkehr nach `CELL`) für die Visualisierungssysteme Gsharp und Tecplot, wobei die Daten für die belegten Maschen um solche für die leeren Maschen ergänzt werden.

Falls Gruppe 19 und `IZ` = 0:
Aufruf von Subroutine `INTGR`. Dort, falls eine Größe der Gruppe übernommen werden soll: Einlesen der Daten und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei.

Falls Gruppe 20 und `IZ` = 0:
Aufruf von Subroutine `TIMEF`. Dort, falls die Gruppe Referenzgruppe für Subgruppe(n) 21 der Eingabe ist: Einlesen der Daten und Abspeichern im Feld `FTIME`.

Falls Subgruppe 21 und `IZ` = 0:
Aufruf von Subroutine `FUNCT`. Dort falls die Gruppe übernommen werden soll: Einlesen der Daten und Abspeichern als „Komponente(n)“ auf eine Hilfsdatei.

(Für das Dateiende des VISART-Files:)

- Falls Dateiende innerhalb eines Rumpf-Pakets: Rücksetzen auf den Stand vor dem Lesen der Zyklus-Gruppe dieses Pakets und weiter nach (9).
  Falls laut Steuereingabe Fortsetzungsfiles vorhanden sind: zurück nach (2).
  Andernfalls: Schließen der jeweiligen VISART-Datei und zurück nach (1).

(Nach dem Lesen aller VISART-Dateien:)

(9) Öffnen der Ausgabedatei(en) und Aufruf der Routinen für das jeweilige Visualisierungssystem:

Falls Eingabeparameter `MODE` = 0: Subroutine `MODE0` (für Werteausgabe).

Falls Eingabeparameter `MODE` = 1: Subroutine `MODE1` (für AVS-FD).

Falls Eingabeparameter `MODE` = $\pm$3: Subroutine `MODE9` (für Gsharp usw.).

Falls Eingabeparameter `MODE` = $\pm$4: Subroutine `MODE4` (für AVS-UCD).

Falls Eingabeparameter `MODE` = $\pm$5: Subroutine `MODE5` (für Tecplot).

- In jeder dieser Routinen werden die Koordinaten des Teilgebiets für die Ausgabe aufbereitet und die Werte der Komponenten darüber von den Hilfsdateien eingelesen und auf die Ausgabedatei(en) geschrieben.

(Programmende.)

5.4 Einige Programmdetails

Zur Visualisierung mit AVS-UCD wird in VISAPTER (im Feld NODE) eine sog. Zellstruktur des Netzes aufgebaut. Die Information hierzu wird den Gruppen 5/15 bzw. 6/16 für das Netzinnere und ggf. für die Netzhülle (im letzteren Fall bei defektiven Netzen auch dem Oberflächennormalenvektor) entnommen. Bei den anderen Visualisierungssystemen wird nur im Falle der Überlagerung (im Feld NODI) eine sog. Überlagerungsstruktur des Netzes aufgebaut. Die Information hierzu wird den Gruppen 5/15 (nur bei solchen ist hier Überlagerung möglich) für das Netzinnere und für die Netzhülle entnommen. Wo die Zell- bzw. Überlagerungsstruktur in VISAPTER aufgebaut werden kann, richtet sich danach, ob die relevanten Gruppen bereits im Kopf-Paket stehen (bei zeitlich konstanter Struktur) oder ob sie in den Rumpf-Paketen stehen (bei zeitlich konstanter oder variabler Struktur). Da der Aufbau der Strukturen u.U. zeitaufwendig ist, kann der optimale Zeitpunkt dafür mittels des Eingabeparameters IRAND (siehe 3.1) festgelegt werden, falls die Redundanz in der VISART-Datei dafür Spielraum läßt.

Bei irregulären Netzen werden in VISAPTER die Netzkoordinaten (im Feld KIND) angesammelt und gespeichert. Die Information hierzu wird den Gruppen 5/15 bzw. Subgruppen 7/17 mit den Netzkoordinaten der einzelnen Lokationen entnommen. Im gleichen Feld werden, auch bei regulären Netzen, die „Löcher" von „schwach-defektiven" Netzen und von expandierten defektiven Netzen vermerkt. Die Information hierzu wird im ersteren Fall den Gruppen 5/15 mit der Angabe der nicht belegten Maschen entnommen, im letzteren Fall bei der Expansion generiert. Wo die Netzkoordinaten von VISAPTER entnommen werden können, richtet sich danach, ob die relevanten Gruppen bereits im Kopf-Paket stehen (bei zeitlich konstanten Koordinaten) oder ob sie in den Rumpf-Paketen stehen (bei zeitlich konstanten oder variablen Koordinaten). Da das Ansammeln der Koordinaten u.U. zeitaufwendig ist, kann der optimale Zeitpunkt dafür mittels des Eingabeparameters IRAKD (siehe 3.1) festgelegt werden, falls die Redundanz in der VISART-Datei dafür Spielraum läßt.

Für das Abspeichern der Komponenten auf Hilfsdateien gilt:
Für jede Komponente (siehe 4.2) werden die Werte über den Koordinaten i, j, k, t (soweit vorhanden) fortlaufend auf temporären Hilfsdateien (ab Dateinummer 51) zwischengespeichert, wobei die erste Koordinate als erste variiert, dann die zweite Koordinate, usw.; bei einer Änderung der t-Koordinate wird ein neuer Satz angefangen. Für AVS-UCD wird auch die temporäre Hilfsdatei Nr. 50 benötigt.

Für das Abspeichern der Indexvektoren auf Hilfsdateien gilt:
Für jeden Indexvektor werden dessen Werte auf temporären Hilfsdateien (von Dateinummer MAXDAT+50 abwärts gezählt) zwischengespeichert.

5.5 Einschränkungen usw.

Die Tabellen für die Eingabeparameter und die Geometrie-abhängigen Felder werden in VISAPTER dynamisch dimensioniert, soweit dies am Programmanfang möglich ist.

Die anderen Felder sind zur Zeit wie folgt dimensioniert:

- Maximal `MAXDAT` = 949 temporäre Hilfsdateien für Komponenten und Indexvektoren.
- Maximal `MAXNT` = 20 Problemzeitintervalle in der Eingabe.
- Maximal `MAXNNO` = 30 auszugebende Größen in der Eingabe, falls `NN`≤ 0 ist.
- Maximal `MAXT` = 5000 auszugebende Problemzyklusnummern/zeitpunkte.
- Maximal `MAXLA1` = 6 unterschiedliche Lokationen der auszugebenden Größen (Lokationen von Größen aus dem Kopf-Paket gelten als verschieden von den gleichen Lokationen der Größen aus dem Rumpf-Paket).
- Maximal `MAXLA0` = 100 auszugebende Problemzyklusnummern/zeitpunkte bei Punktgrößen (Partikel).
- Maximal `MAXMU` = 3 Teilnetze in den VISART-Dateien.
- Maximal `MAXNU` = 2 (zu leeren Gruppen in den Rumpf-Paketen) komplementäre Gruppen im Kopf-Paket.
- Maximal `MAXND` = 2 VISART-Dateien (die Anzahl der jeweiligen Fortsetzungsfiles ist jedoch unbegrenzt).
- Maximal `MAXWAL` = 40 innere Wände (nur für AVS-UCD mit Überlagerung).
- Maximal `MAXCLB` = 24 Zeichen in den Labels (einschließlich später entfernter Leerzeichen).

Einige andere Felder sind mit den dynamisch dimensionierten gekoppelt:

- Maximal `MAXNOD` = max(`M`, 1000) „Nodes“ insgesamt (nur für Anlage einer Zellstruktur—siehe 5.4), und maximal `MAXINT` = max(`M`, 1000) interpolierte „Nodes“ (nur für AVS-UCD mit Überlagerung), wo `M` ungefähr viermal die Anzahl der Maschen des Netzes ist.
- Maximal `MXFT` = min(`MAXIJK` $*$ 6, 100000) Problemzeitpunkte in Zeitfunktionsgruppen und 2mal die Anzahl der Werte in einer Integralgrößen-Gruppe, wo `MAXIJK` ungefähr dreimal die Anzahl der Maschen des Netzes ist (mindestens 50000).

Bei der derzeitigen Implementierung von VISAPTER gilt für die Indikatoren (siehe [1b], Tabellen 4.3.4, ..., 4.3.7) der VISART-Dateien:

- Es können nur Maschennetze (`IZGEO` = 1 oder = 3) behandelt werden.
- Für reguläre Netze (`IZGEO` = 1) können nur kartesische und polare bzw. zylindrische Koordinatensysteme `IZSYS` = 200, 210, 220, 202, 212, 300, 302 behandelt werden.
- Für irreguläre Netze (`IZGEO` = 3) können nur kartesische Koordinatensysteme `IZSYS` = 200, 210, 220, 300 behandelt werden.
- Es können nur die „üblichen“ Reihenfolgen `IxORD` = 1 bzw. = 12 bzw. = 123 behandelt werden.
- Es können alle in der Tab. 4.3.6 von [1b] explizit aufgeführten Lokationen `IxLOC` behandelt werden (wo ± 33, ± 77 nicht für volle Netze, ± 88 nicht für defektive Netze sinnvoll sind).

Es können nur „einfache“, „nicht-degenerierte“ Netzhüllen behandelt werden.

Die Alleindarstellung von Oberflächen ist für Netzgrößen, mit Lokationen -11, -22, -44, -88, bis jetzt noch nicht möglich.

Die Darstellung eindimensionaler Subgebiete im dreidimensionalen Raum ist bis jetzt noch nicht möglich.

Für Punktgrößen sind bis jetzt noch keine Teilgebiete auswählbar.

Die folgenden Optionen sind bis jetzt nur für das Visualisierungssystem Tecplot implementiert:

- Teilnetze (siehe [1a], [1b]) (nicht zusammen mit Flächenscharen!)
- Scatter-Plots (siehe 2.4).
- Flächenscharen im dreidimensionalen Raum (siehe 3.2) (nicht zusammen mit Teilnetzen!)
- Cell-based Darstellung auf zweidimensionalen Subgebieten entlang der Maschenhüllen dreidimensionaler Netze (siehe 2.3).
- Binäre Ausgabedateien (siehe 5.1).

Die folgenden Optionen sind bis jetzt nur für das Visualisierungssystem AVS/Express mit UCD implementiert:

- Berücksichtigung innerer Wände.

5.6 Aufbau der Benutzeroberfläche

Die grafische Benutzeroberfläche für die VISART-Dienste unter UNIX wurde bereits im Teil 1 der VISART-Dokumentation [1a] vorgestellt. Sie ist in der Skriptsprache Tcl/Tk implementiert und setzt die Installierung des entsprechenden Interpreters voraus, der u.a. die Tk-Funktionen mit der Xlib-Library realisiert.

Die Tcl/Tk-Skripte der VISART-Benutzeroberfläche stehen unter DCE/DFS im Verzeichnis

`/fzk/inr/@sys/VISART/tcltk`

Der Aufruf des Skripts `visart` öffnet das Hauptfenster der Benutzeroberfläche. Von dort gelangt man durch Anklicken der Tasten „VISAPTER“ oder „VISAPTER(0)“ in die Skripte `visapter.tcl` bzw. `visapter0.tcl`, die die entsprechenden Fenster (siehe 3.4) öffnen und ggf. in die Skripte für weitere Unterfenster verzweigen. Beim Aufruf der Visualisierungssysteme aus dem VISAPTER-Fenster werden mit den Skripten `visuvxp`, `visucom3` oder `visutcplt` (für AVS/Express bzw. Gsharp bzw. Tecplot-7) entsprechende Fenster geöffnet, wo die Dateien für die/das gewünschte AVS-Application bzw. Gsharp-Script bzw. Tecplot-Macro spezifiziert werden (siehe 6.2.3, 6.3.3, bzw. 6.4.3). Das Skript `visualize.tcl` übergibt diese Datei zusammen mit der Ausgabedatei von VISAPTER an das gerufene Visualisierungssystem.

6. Spezifische Anwendungen des Programms

In diesem Kapitel werden Besonderheiten der Ein- und Ausgabe für die verschiedenen Visualisierungssysteme (bzw. in 6.1 für die Sonderfunktion der Werteausgabe) beschrieben.

6.1 Anschauen von Zahlenwerten

6.1.1 Besonderheiten

Mit VISAPTER können u.a. die Werte von Größen über beliebigen räumlichen und zeitlichen Dimensionen zur Inspektion ausgegeben werden. Die Werte skalarer Größen bzw. die Werte der Komponenten vektorieller Größen werden in einer Zeile mit den Maschenindizes und den Koordinatenwerten angezeigt, wobei die Indizes und Koordinaten über das spezifizierte Netz/Teilgebiet und das spezifizierte Problemzeitintervall laufen.

`MODE` $= 0$ wählt die Werteinspektion.

Es werden nur die Werte belegter Maschen des Netzes ausgedruckt, unter Berücksichtigung von in der VISART-Datei definierten Teilnetzen, schwach oder stark defektiven Netzen.

`IZ` darf alle Werte von -3 bis 3 annehmen. Für `IZ` < 0 ist auch mehr als ein Problemzeitpunkt zulässig. Bei negativen Werten von `IZ` werden die Maschenwerte über dem belegten Netz für jeden Problemzeitpunkt gesondert ausgedruckt, mit der Problemzeit als Überschrift. Andernfalls wird die Problemzeit als Koordinate, ggf. zusammen mit den räumlichen Koordinaten, ausgedruckt. Wenn die Dimension des auszudruckenden Netzes/Teilgebiets aus den Maschenindizes der Größen entnommen werden soll, genügt die Angabe von `IZ` $= -9$, $= 0$ oder $= 9$ für nur räumliche Dimensionen bzw. nur die zeitliche Dimension bzw. räumliche und zeitliche Dimensionen.

Es können nur Werte im Innern des Netzes ausgedruckt werden. Die ausgedruckten Maschenindizes beziehen sich auf das zugrunde liegende Netz. Die ausgedruckten Koordinaten und ggf. die Vektorkomponenten beziehen sich, wenn `IKRUM` $\neq 0$, auf das in der Umrechnung gewählte Koordinatensystem. Es sind Größen des Typs `REAL` und `INTEGER` erlaubt. Für Größen der Gruppen 5/15 und Subgruppen 7/17 wird für jeden Problemzeitpunkt auch das Minimum und das Maximum der Werte der Größen bzw. der Größenkomponenten über den belegten Maschen des Netzes und ihrer absoluten Beträge berechnet und ausgegeben.

6.2 Visualisierung mit AVS/Express

6.2.1 Besonderheiten

Die Ausgabe der Daten für AVS ist im Field-Data-Format (FD-Format) oder im Unstructured-Cell-Data-Format (UCD-Format) möglich. Für UCD steht eine andere und mächtigere Menge von AVS-Moduln zur Visualisierung zur Verfügung als für FD. Vor allem erlaubt UCD die Definition von defektiven Netzen, wie sie von den Gruppen 6/16 in der VISART-Datei beschrieben werden, durch die explizite Spezifizierung der belegten Maschen. Auch Gruppen 5/15 für Teilnetze oder für schwach defektive Netze, definiert durch Maschenbelegungsgruppen (siehe 4.1) in der VISART-Datei, lassen sich mit UCD auf diese Weise visualisieren. Mit FD können nur Gruppen 5/15 für volle Netze/Teilgebiete visualisiert werden.

`MODE` $= 1$ wählt die Ausgabe für AVS im FD-Format; `MODE` $= \pm 4$ wählt die Ausgabe für AVS im UCD-Format, und zwar mit positivem Vorzeichen eine node-based Darstellung, mit negativem Vorzeichen eine cell-based Darstellung. Cell-based Darstellungen lassen sich zwei- und dreidimensional sinnvoll nur durch Konturplots (dreidimensional evtl. mit Schnitten) repräsentieren, wofür es spezielle AVS-UCD-Moduln gibt. FD-Dateien müssen unter AVS/Express mit dem Modul `Read Field` eingelesen werden, UCD-Dateien mit dem Modul `Read UCD`.

`IZ` kann alle Werte $\neq 0$ annehmen, für FD auch den Wert 0. Für `IZ` < 0 ist auch mehr als ein Problemzeitpunkt zulässig. Konventionelle Funktionsgrafik scheint mit den AVS-UCD-Moduln nicht möglich zu sein; alle Darstellungen, auch von zwei- oder eindimensionalen Netzen/Teilgebieten finden im dreidimensionalen Bildraum statt.

Die zugrunde liegenden Netze/Teilgebiete können kartesisch, krummlinig orthogonal oder irregulär sein. Bei krummlinigen Koordinatensystemen müssen Vektorkomponenten und Koordinaten auf kartesische Koordinaten umgerechnet werden (`IKRUM` $= 1$ oder `IKRUM` $= 2$).

Außer bei `IZ` $= 0$ und (für AVS-FD) $= -1$ können skalare und/oder vektorielle Größen, des gleichen Typs (`REAL` oder `INTEGER`) und über dem gleichen Netz/Teilgebiet, mit gleichen Lokationen, ausgewählt werden. Bei `IZ` $= 0$ oder (für AVS-FD) $= -1$ können nur skalare Größen oder Komponenten vektorieller Größen ausgewählt werden.

6.2.2 Aufbau der AVS-Dateien

Für `MODE` = 1 erstellt VISAPTER die für AVS aufbereiteten Daten im FD-Format. Im hier verwendeten „Data-Parsing Input Mode“ werden zwei ASCII-Files[1] auf den Einheiten Nr. 20 und 21 erstellt (das hier verwendete FD-Format ist im Chapter 2 von [3a] beschrieben, die Erweiterungen für "Time Dependent (Multistep) Data“ in [3c]). Einer der Files, der Description-File (Nr. 21, Default-Suffix `.fld`), dient als Header für den anderen File, den Data-File (Nr. 20, Default-Suffix `.data`), der die Werte der Größen zusammen mit den Koordinaten enthält.

Für `MODE` = ±4 erstellt VISAPTER die für AVS aufbereiteten Daten im UCD-Format. Es wird ein ASCII-File[1] auf der Einheit Nr. 20 (Default-Suffix `.inp`) erstellt (das hier verwendete UCD-Format ist im Chapter 2 von [3a] beschrieben, die Erweiterungen für "Time Dependent (Multistep) Data“ in [3c]).

Die Daten der Files werden durch die Aufteilung auf Zeilen und mit Hilfe von Schlüsselworten und Schlüsselparametern gegliedert. Letztere werden in den folgenden Beschreibungen in Schreibmaschinenschrift angegeben, während Variable kursiv geschrieben werden.

In den von VISAPTER erzeugten Labels der Komponenten werden für die Ausgabe auf die AVS-Files At-Zeichen, Klammern, Sterne und Kommata (siehe 4.2) durch Unterstriche ersetzt. Für AVS-FD werden bei Vektoren noch Indizes zur Unterscheidung der Vektorkomponenten angehängt.

[1] Eine `BINARY`-Option (siehe 5.1) ist für AVS/Express nicht implementiert.

Aufbau des FD-Description-Files

Die folgende Anordnung gilt für die räumliche Visualisierung, d.h. für effektive Dimensionen $d \geq 2$ oder für Raumdimensionen $D \geq 2$. Bei rechteckigen Netzen wird die Anordnung über `rectilinear` Koordinaten verwendet, bei krummlinigen orthogonalen Netzen über `irregular` Koordinaten [3a]. Die in eckigen Klammern stehenden Zeilen 8, 9 und 12 werden nur geschrieben, wenn bei `IZ` < 0 mehr als *ein* Problemzeitpunkt ausgegeben werden soll [3b].

1. `NDIM =` d
2.a `DIM1 =` $\bar{i}_1$
2.b [`DIM2 =` $\bar{i}_2$]
2.c [`DIM3 =` $\bar{i}_3$]
3. `NSPACE =` D
4. `VECLEN =` n
5. `DATA = { FLOAT | INTEGER }`
6. `FIELD = { RECTILINEAR | IRREGULAR }`

Für jede Komponente $\nu = 1, \ldots, n$ die folgende Zeile 7:

7. `LABEL =` *Label*$_\nu$
8. [`NSTEP =` T]

Für jeden Problemzeitpunkt $\tau = 1, \ldots, T$ die folgenden Zeilen 9 bis 12:

9. [`TIME VALUE =` t_τ]

Für jede Komponente $\nu = 1, \ldots, n$ die folgende Zeile 10:

10. `VARIABLE` ν `FILE =` *FD-Data-File* `FILETYPE = ASCII SKIP =` s_ν^τ

Für jede Koordinate $\delta = 1, \ldots, d$ die folgende Zeile 11:

11. `COORD` δ `FILE =` *FD-Data-File* `FILETYPE = ASCII SKIP =` s_δ
12. [`EOT`]

$\bar{i}_{1\ldots d}$ sind die Anzahl der Lokationen des Teilnetzes in den Koordinatenrichtungen bzw. die Anzahl der Problemzeitpunkte (bei `IZ` > 0).
d ist die effektive Dimension der Darstellungskoordinaten;
D ist die Raumdimension;
n ist die Anzahl der Komponenten (siehe 4.2);
T ist die Anzahl der Problemzeitpunkte, falls `IZ` < 0 ist; andernfalls 1;
s_ν^τ ist die Anzahl der Zeilen im FD-Data-File, nach deren Überspringen die Daten der Komponente ν für den Problemzeitpunkt τ stehen;
s_δ ist die Anzahl der Zeilen im FD-Data-File, nach deren Überspringen die Daten der Koordinate δ stehen.

Die folgende Anordnung gilt für die Darstellung als Funktionsgraph, d.h. für IZ $= -1$ mit $D = 1$ oder IZ $= 0$. Für IZ $= -1$ ist nur *ein* Problemzeitpunkt möglich.

1. NDIM = 2

2.a DIM1 = 2

2.b DIM2 = $\bar{i}$

3. NSPACE = 2

4. VECLEN = n

5. DATA = { FLOAT | INTEGER }

6. FIELD = RECTILINEAR

Für jede Komponente $\nu = 1, \ldots, n$ die folgende Zeile 7:

7. LABEL = *Label*$_\nu$

Für jede Komponente $\nu = 1, \ldots, n$ die folgende Zeile 10:

10. VARIABLE ν FILE = *FD-Data-File* FILETYPE = ASCII SKIP = s_ν

Für jede Koordinate $\delta = 1, 2$ die folgende Zeile 11:

11. COORD δ FILE = *FD-Data-File* FILETYPE = ASCII SKIP = s_δ

$\bar{i}$ ist die Anzahl der Funktionswerte, d.h. der Lokationen bzw. der Problemzeitpunkte, falls IZ $= 0$.

Aufbau des FD-Data-Files

Der Data-File enthält Daten im Format (8G11.3), d.h. acht Real- oder Integer-Zahlen pro Zeile.

Bei der Ausgabe von Werten für die räumliche Visualisierung werden für jede Komponente die Werte, beginnend mit einer neuen Zeile, über den Koordinaten—i, j, k für drei räumliche Dimensionen, bzw. i, j, t für zwei Dimensionen und mehrere Zeitpunkte, bzw. i, j für zwei Dimensionen und *einen* Zeitpunkt, usw.—fortlaufend geschrieben, wobei die erste Koordinate als erste variiert, dann die zweite Koordinate, dann die dritte. Bei Vektoren über krummlinigen Koordinaten werden die auf kartesische Koordinaten umgerechneten Vektorkomponenten geschrieben. Anschließend folgen die Koordinaten selbst, in der obigen Reihenfolge, jeweils beginnend mit einer neuen Zeile. Bei rechteckigen Koordinaten des Netzes oder Teilgebiets werden nur die Koordinaten der Koordinatenflächen ausgegeben (falls der Eingabeparameter KNULL $\neq 0$ ist, werden für die damit bezeichnete Koordinate Nullen statt der Koordinatenwerte ausgegeben); bei krummlinigen Koordinaten werden die kartesischen Koordinaten jedes Maschenpunktes ausgegeben.

Bei der Ausgabe von Werten für die Darstellung als Funktionsgraph werden die Funktionswerte und die Problemzeit bzw. die Koordinate (abwechselnd) über fiktiven zweidimensionalen rechteckigen Koordinaten geschrieben. Anschließend folgen als fiktive Koordinaten Nullen, und zwar zuerst zwei in der einen Koordinatenrichtung, dann soviel, wie die Anzahl der Funktionswerte beträgt, jeweils beginnend mit einer neuen Zeile.

Aufbau des UCD-Files

Die folgende Anordnung gilt nur für die räumliche Visualisierung in dreidimensionalen Darstellungskoordinaten (`IZ` $\neq 0$). Als UCD-Cells werden bei 3 (räumlichen oder zeitlichen) Dimensionen Hexaeder, bei 2 (räumlichen oder zeitlichen) Dimensionen Vierecke, bei 1 Dimension Strecken verwendet.

1. T
2. `data`

Für jeden Problemzeitpunkt $\tau = 1, \ldots, T$ die folgenden Zeilen 3 bis 10/10':

3. `step`τ [t=t_τ]
4. *Anzahl der Nodes* *Anzahl der Cells*

Für jeden Node die folgende Zeile 5:

5. *Node-Nummer* $Koordinate_1$ $Koordinate_2$ $Koordinate_3$

Für jede Cell die folgende Zeile 6:

6. *Cell-Nummer* 1 { `hex` | `quad` | `line` } *Node-Nummern der Cell-Ecken*

Falls node-based: Zeile 7; falls cell-based: Zeile 7':

7. n 0

7'. 0 n

8. N c_1 ... c_N

Für jede Größe $\gamma = 1, \ldots, N$ die folgende Zeile 9:

9. $Label_\gamma$, `SI`

Falls node-based: für jeden Node die folgende Zeile 10:

10. *Node-Nummer* $Komponente_1^\tau$ ... $Komponente_n^\tau$

Falls cell-based: für jede Cell die folgende Zeile 10':

10'. *Cell-Nummer* $Komponente_1^\tau$ ... $Komponente_n^\tau$

T ist die Anzahl der Problemzeitpunkte, falls `IZ` < 0 ist; andernfalls 1;
N ist die Anzahl der Größen;
c_γ ist die Anzahl der Vektorkomponenten der Größe γ;
n ist die Anzahl der Komponenten (siehe 4.2), wobei

$$n = \sum_{\gamma=1}^{N} c_\gamma$$

Bei krummlinigen Koordinaten werden die kartesischen Koordinaten jedes Maschenpunktes ausgegeben. Falls der Eingabeparameter `KNULL` $\neq 0$ ist, werden für die damit bezeichnete Koordinate Nullen statt der Koordinatenwerte ausgegeben. Bei Vektoren über krummlinigen Koordinaten werden die auf kartesische Koordinaten umgerechneten Vektorkomponenten geschrieben.

6.2.3 AVS/Express-Aufruf

Die von VISAPTER für AVS in den obigen Formaten erstellten Dateien können von AVS eingelesen und interpretiert werden. Leider kann AVS/Express beim Aufruf nicht der Name der Datei oder des Dateiverzeichnisses übergeben werden; dieser muß unter der AVS/Express-Sitzung im Einlesemodul eingestellt werden (bei AVS-FD nur der Name des Description-Files, weil dort der Pfad des Data-Files eingetragen steht). Jedoch kann AVS/Express der Name eines die Visualisierung steuernden Application-Files als Parameter beim Aufruf übergeben werden[1].

In Ergänzung zu VISAPTER wird eine Anzahl von AVS/Express-Applications, sowohl für AVS-FD, als auch für AVS-UCD, zur Verfügung gestellt, die unter dem Visualisierungssystem die Datei(en) einlesen und den Aufbau einer dem Fall angemessenen Standarddarstellung steuern. Der Benutzer muß lediglich die dem jeweiligen Problem angemessene Application auswählen, z.B. danach, ob es sich um Skalar- oder Vektorgrößen, Netze ohne oder mit Löchern, handelt, und er muß entscheiden, welche Repräsentation er für die Größen wünscht. Unter DCE/DFS stehen die Application-Files in den Verzeichnissen

`/fzk/inr/@sys/VISART/vxp-networks/vxp-field` (für AVS-FD)
`/fzk/inr/@sys/VISART/vxp-networks/vxp-ucd` (für AVS-UCD)

unter den Dateinamen
vxp$ijk\ell mn$.v

wo i, j, k, ℓ, m, n für Ziffern stehen, deren Werte die folgenden Parameter verschlüsseln:

- i Bilddimension (1, 2 oder 3)
- j Anzahl der zusammen zu visualisierenden Größen (1, ...)
- k node-based (1) oder cell-based (2) Darstellung
- ℓ Netz-Skalar (1), Netz-Vektor (2), Netz-Skalar mit -Vektor (3), Partikel-Skalar (4), Partikel-Vektor (5); Integral-Skalar (0)
- m Repräsentation der Werte der darzustellenden Größen (in Abhängigkeit von i und ℓ):
  (1) Konturplot für Skalare, Pfeile für Vektoren, auf Schnitten durch 3D-Netze
  (2) Höhenlinienplot für Skalar, Stromlinien für Vektoren, Symboldurchmesser für Partikel
  (3) Konturplot für Skalare, Pfeile für Vektoren, Symbolfarbe bzw. Vektorpfeile für Partikel
  (4) dreidimensionaler Plot für 2D-Skalare, Iso-Flächen für 3D-Skalare
  (5) für 2 Skalare: dreidimensionaler Plot mit farbigen Konturen
  (6) für 1 Skalar und 1 Vektor: Konturplot mit Vektorpfeilen
  (7) für 1 Skalar und 1 Vektor: Konturplot mit Stromlinien
- n Einzelbild (1) oder Bildfolge für die Verfilmung (2) oder Animation (3)

Beispiel: der Application-File `vxp211131.v` erzeugt einen Konturplot in node-based Darstellung einer Skalargröße über einem zweidimensionalen Netz bzw. Gebiet.

AVS/Express wird hier mit dem Kommando

`vxp -nohw -ne` *Application-File*

aufgerufen. Anschließend kann der Benutzer die Steuerung von AVS/Express selbst übernehmen.

Das Visualisierungssystem AVS/Express kann auch über die grafische Benutzeroberfläche [1a] aufgerufen werden. Im Anschluß an das „Ausführen“ des Programms VISAPTER über das VISAPTER-Fenster (Bild 3) der grafischen Benutzeroberfläche wird ein Unterfenster (Bild 4) zum Aufruf von AVS/Express angeboten. Alternativ kann das Unterfenster auch durch Betätigen der Taste „Visualisieren“ im VISAPTER-Fenster eröffnet werden, wenn die AVS-Datei(en) bereits vorhanden ist/sind.

[1] „Application“ bei AVS/Express ist gleichbedeutend mit „Netzwerk“ unter AVS.

Im AVS/Express-Unterfenster kann der Benutzer unter den vorhandenen Application-Files für die Standarddarstellungen auswählen, indem er mit den Menütasten die Kombination der in Frage kommenden oder gewünschten Parameter einstellt (entsprechend den Stellen ℓ, m, n im Dateinamen des gewählten Macro-Files; die Werte der Stellen i, j, k werden aus den Einstellungen im VISAPTER-Fenster übernommen).

Anstelle eines Standard-Macro-Files kann der Benutzer auch andere, z.B. selbst erstellte bzw. modifizierte Application-Files spezifizieren.

Durch Betätigen der Taste „Visualisierung“ wird AVS/Express über das obige Kommando aufgerufen. Nach Beendigung der AVS/Express-Sitzung wird in das VISAPTER-Fenster der Benutzeroberfläche zurückgekehrt.

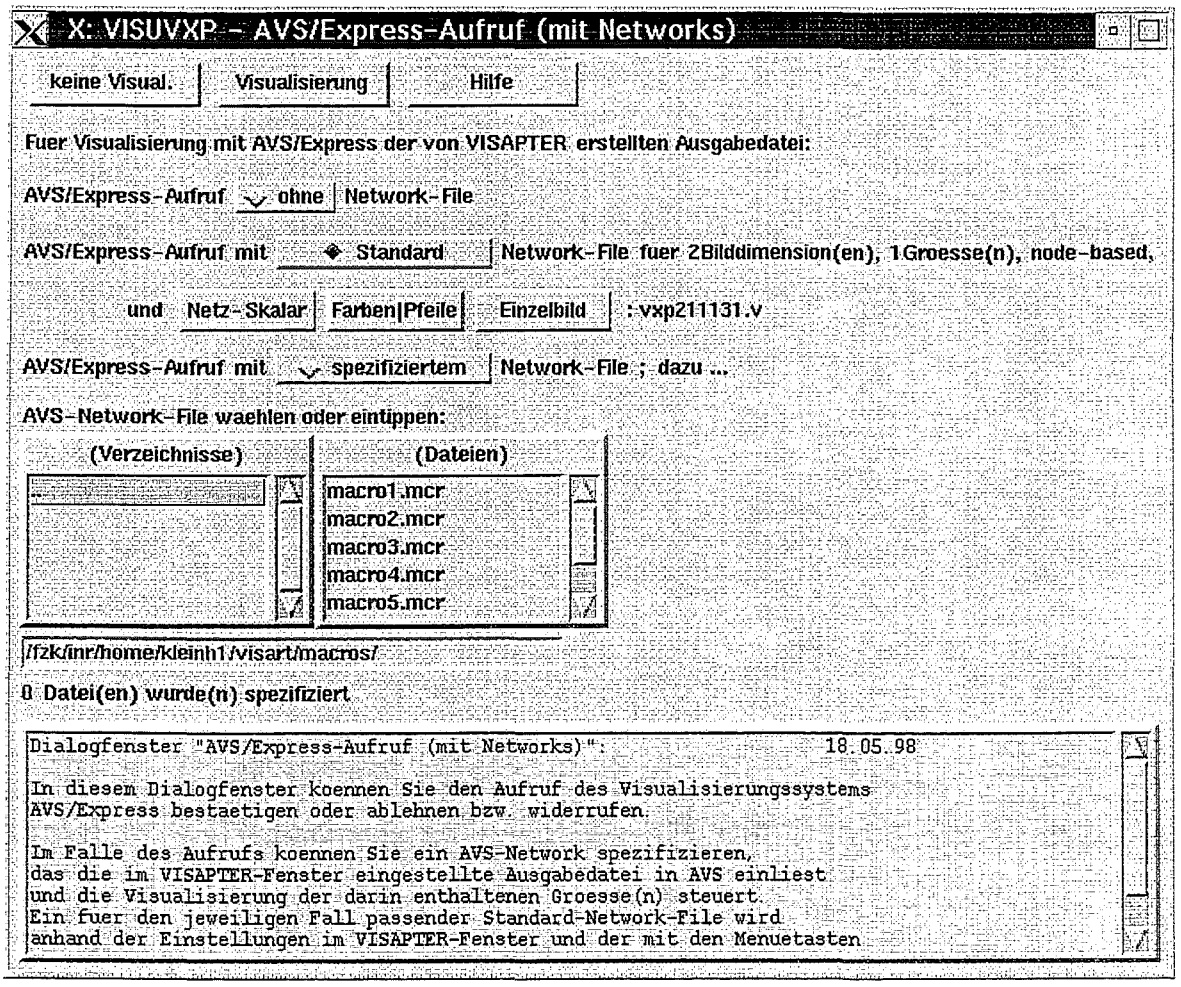


Bild 4: Visualisierungs-Fenster für AVS/Express

6.2.4 Beispiele

Bild 4.a und Bild 4.b zeigen Beispiele für die Visualisierung von Daten im VISART-Postprocessor-Format mit AVS/Express-UCD. Die zugehörigen Steuereingaben für das Programm VISAPTER stehen in den Listen 4.a und 4.b.

Beide Beispiele stammen aus dem FLUTAN-Code. Beispiel 4.a für das sog. RAMONA-Problem beruht auf einem dreidimensionalen, regulären, defektiven $9 \times 11 \times 13$ -Netz in Zylindergeometrie. Dargestellt wird die Temperatur, Gruppe `SITL`, als cell-based Konturplot. Das Bild 4.a wird mit der AVS/Express-Application `vxp312131.v` (unter Benutzung der AVS/Express-Moduln `Read UCD`, `external edges` und `Uviewer3D`) aus der von VISAPTER mit der Eingabe aus Liste 4.a erzeugten Datei produziert.

Beispiel 4.b für das sog. UMLENK-Problem beruht auf einem dreidimensionalen, regulären, defektiven $24 \times 20 \times 20$ -Netz in kartesischer Geometrie. Dargestellt wird die Geschwindigkeit, als Überlagerung der Gruppen `SIVEL` und `SSVEL`, als (node-based) Vektorplot. Das Bild 4.b wird mit der AVS/Express-Application `vxp311231.v` (unter Benutzung der AVS/Express-Moduln `Read UCD`, `glyph`, `Arrow1`, `external edges` und `Uviewer3D`) aus der von VISAPTER mit der Eingabe aus Liste 4.b erzeugten Datei produziert. (Die Strömung tritt durch den rechten Stutzen ein, wird im Kasten duch eine Trennwand umgelenkt und tritt durch den linken Stutzen aus[1].)

Liste 4.a: VISAPTER-Steuereingabe zu Bild 4.a

```
-4 -3  1  0  0  4  4
1  1  99  1
17  'SITL'  'SI INDEX'  0  -1 -1  -1 -1  -1 -1
```

Liste 4.b: VISAPTER-Steuereingabe zu Bild 4.b

```
4 -3  0  0 -2  4  4
'SI'  'SS'  'SS NORMV'
1  1  99  2
17  'SIVEL'  'SI INDEX'  0  -1 -1  -1 -1  -1 -1
17  'SSVEL'  'SS INDEX'  0  -1 -1  -1 -1  -1 -1
```

[1] Leider geht beim Ausdrucken ein Teil der Qualität des Bildes verloren. Die überflüssigen Umrandungen dort, wo die Stutzen am Kasten ansetzen, rühren von einer Unvollkommenheit der AVS-Moduln her.

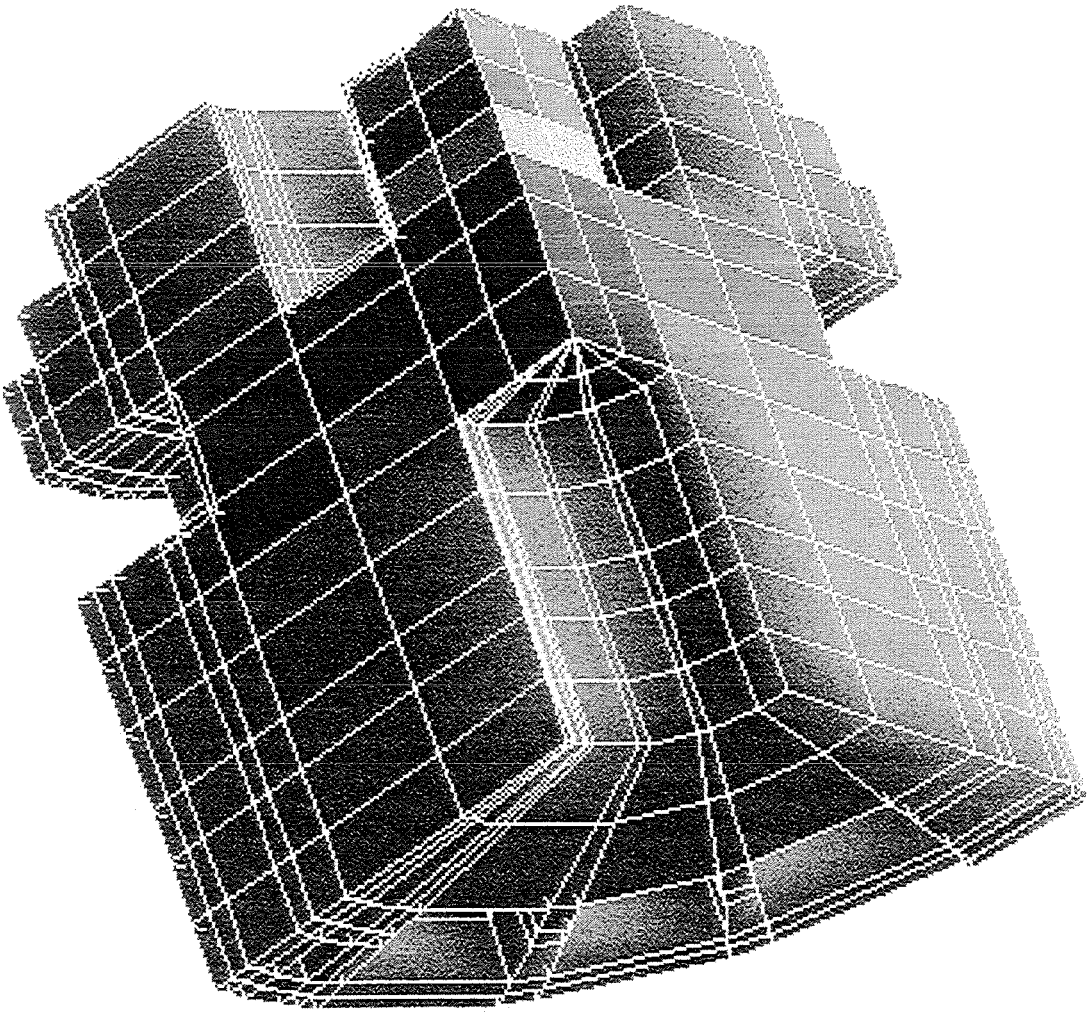


Bild 4.a: RAMONA-Problem aus dem FLUTAN-Code

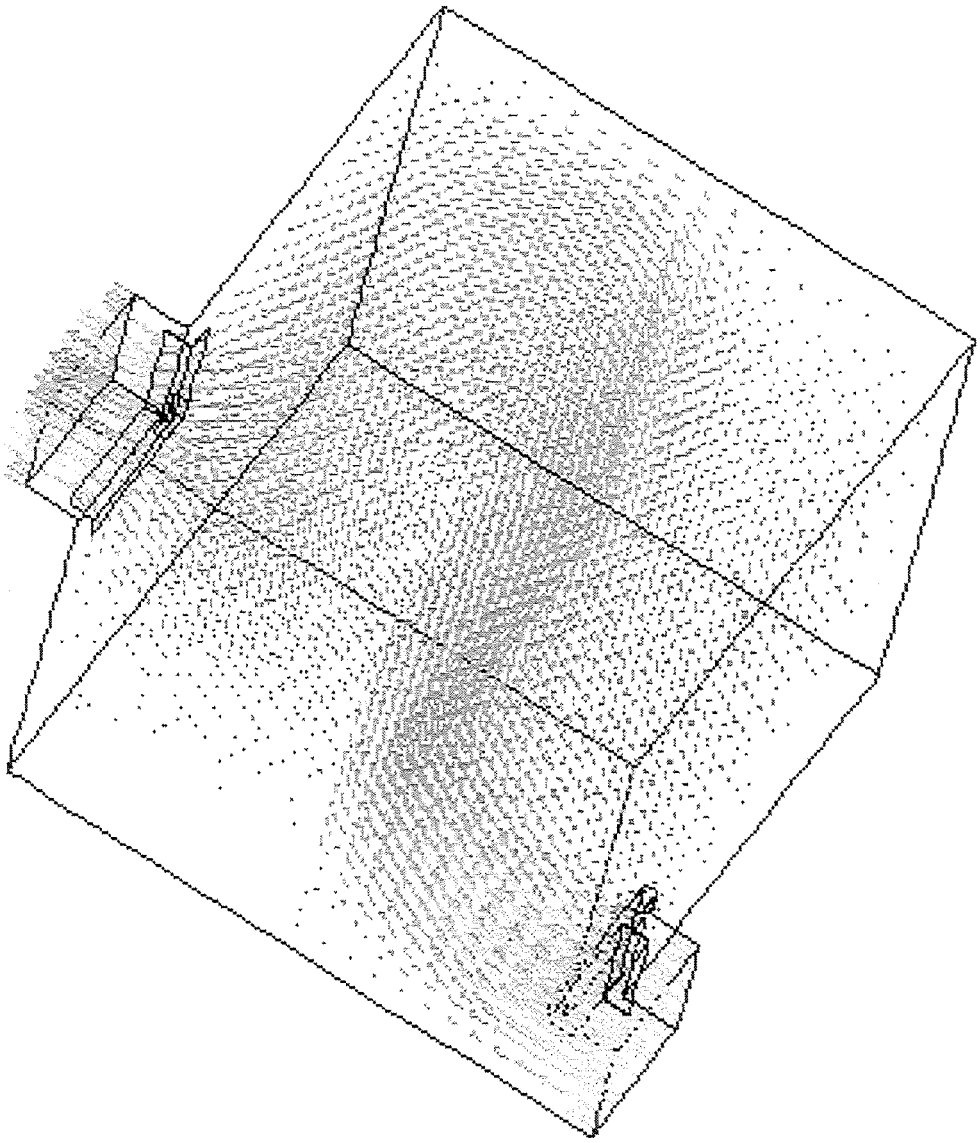


Bild 4.b: UMLENK-Problem aus dem FLUTAN-Code

6.3 Visualisierung mit Gsharp

6.3.1 Besonderheiten

`MODE` = ± 3 wählt die Ausgabe für Gsharp [4a], und zwar mit positivem Vorzeichen eine node-based Darstellung, mit negativem Vorzeichen eine cell-based Darstellung. Cell-based Darstellungen lassen sich zwei- und dreidimensional sinnvoll nur durch Konturplots (dreidimensional evtl. mit Schnitten) repräsentieren, wobei in Gsharp der „Graph Type" `Grid` oder `3DGrid` verwendet wird.

Teilnetze in der VISART-Datei kann VISAPTER nicht für Gsharp aufbereiten. Für schwach defektive Netze, definiert durch Maschenbelegungsgruppen (siehe 4.1) in der VISART-Datei, übergibt VISAPTER als „undefined values" für die nicht belegten Maschen Fragezeichen (`?`) statt numerischer Werte. Stark defektive Netze, definiert durch Gruppen 6/16 in der VISART-Datei, expandiert VISAPTER für Gsharp auf volle Netze mit Löchern und verfährt dann wie für schwach defektive Netze.

`IZ` darf alle Werte -3, ..., 2 annehmen. Für `IZ` < 0 ist auch mehr als ein Problemzeitpunkt zulässig.

Die zugrunde liegenden Netze/Teilgebiete können kartesisch oder polar sein.

Bei vektoriellen Größen können nur die im Teilgebiet liegenden Vektorkomponenten dargestellt werden.

Es können mehrere Größen, auch skalar und vektoriell gemischt, über dem gleichen Netz/Teilgebiet, nicht unbedingt mit gleichen Lokationen und daher nicht unbedingt mit gleicher Anzahl von Koordinatenwerten, ausgewählt werden. Es sind Größen des Typs `REAL` und `INTEGER`, auch gemischt, erlaubt (letztere werden auf erstere umdefiniert).

6.3.2 Aufbau der Datei für Gsharp

Gsharp hat, im Gegensatz zu AVS und Tecplot, kein festgelegtes Format für die Eingabedatei. Das System erwartet lediglich, daß die Eingabedaten als ASCII-Daten in einer tabellarisch geordneten Reihenfolge angeliefert werden. VISAPTER erstellt die Datei für Gsharp in einem Format, das von einem dazu kompatiblen Interpretationsprogramm (siehe 6.3.3) leicht verarbeitet werden kann. Das Format der Datei (Default-Suffix `.dat`) sieht node-based und cell-based Darstellung vor, im ersteren Fall mit beliebigen Lokationen der Größen, sowie auch krummlinige und irreguläre Netze, die von Gsharp bisher nicht darstellbar sind. Dadurch ist die Datei auch zur Datenübergabe an Visualisierungssysteme auf PCs, z.B. SURFER [4b], geeignet.

In der folgenden Beschreibung entspricht jede Zeile mindestens *einer* Zeile der Ausgabedatei. Falls nichts anderes angegeben ist, bezeichnen mit `I,...,N` beginnende Namen `INTEGER`-Variable im Format `I8`, mit `C` beginnende Namen `CHARACTER`-Variable im Format `A8`, alle andere Namen `REAL`-Variable im Format `E16.8`. (Bei Netzen mit „Löchern" stehen in den Komponenten $S_{\tau\lambda}$ an den nicht belegten Maschen statt numerischer Werte Fragezeichen (`?`), woran Gsharp erkennt, daß die Masche nicht belegt ist.) Eine Zeile enthält maximal 80 Zeichen; für jede Zeile der Beschreibung werden so viele Zeilen auf die Datei geschrieben, wie durch die Aufteilung der Variablen auf die Zeilenlänge benötigt werden.

1. `CCNAME,CCRELS,CCAUTH,CCDATE,CCTIME`
2. `CJNAME,CJNUMB,CJAUTH,CJDATE,CJTIME`
3. `CINAME,CINUMB,CIAUTH,CIDATE,CITIME`
4. `CIIDE1(1),...,CIIDE1(10)`
5. `CIIDE2(1),...,CIIDE2(10)`
6. `ISYS,IZ,NR,NS,IR,JR,KR,IRJRKR,LN,LC`
7. `CI,CJ,CK`
8. `XR(1),...,XR(IR)` bzw., wenn `IRJRKR` $\neq 0$, `XR(1),...,XR(IRJRKR)`

(wenn `NS` > 1:)

9. `YR(1),...,YR(JR)` bzw., wenn `IRJRKR` $\neq 0$, `YR(1),...,YR(IRJRKR)`

(wenn `NS` > 2:)

10. `ZR(1),...,ZR(KR)` bzw., wenn `IRJRKR` $\neq 0$, `ZR(1),...,ZR(IRJRKR)`
11. `KT,KS,KL`

(wenn `KT` > 0:

12. `TK(1),...,TK(KT)`
13. `CS1(1),CS2(1),...,CS1(KS),CS2(KS)`
14. `KK(1),...,KK(KS)`
15. `LL(1),...,LL(KS)`

(wenn `KL` > 0: für $\alpha = 1 \ldots$`KL` Lokationsanordnungen folgt jeweils 16 bis 19:)

16. $\texttt{IA}_\alpha,\texttt{JA}_\alpha,\texttt{KA}_\alpha,\texttt{IAJAKA}_\alpha,\texttt{LA}_\alpha$
17. $\texttt{XA(1)}_\alpha,\ldots,\texttt{XA(IA}_\alpha\texttt{)}_\alpha$ bzw., wenn $\texttt{IAJAKA}_\alpha \neq 0$, $\texttt{XA(1)}_\alpha,\ldots,\texttt{XA(IAJAKA}_\alpha\texttt{)}_\alpha$

(wenn `NS` > 1:)

18. $\texttt{YA(1)}_\alpha,\ldots,\texttt{YA(JA}_\alpha\texttt{)}_\alpha$ bzw., wenn $\texttt{IAJAKA}_\alpha \neq 0$, $\texttt{YA(1)}_\alpha,\ldots,\texttt{YA(IAJAKA}_\alpha\texttt{)}_\alpha$

(wenn `NS` > 2:)

19. $\texttt{ZA(1)}_\alpha,\ldots,\texttt{ZA(KA}_\alpha\texttt{)}_\alpha$ bzw., wenn $\texttt{IAJAKA}_\alpha \neq 0$, $\texttt{ZA(1)}_\alpha,\ldots,\texttt{ZA(IAJAKA}_\alpha\texttt{)}_\alpha$

(für $\tau = 1 \ldots$`KT` Problemzeitpunkte folgt jeweils...
...für $\lambda = 1 \ldots$`KS` Komponenten:)

20. $\texttt{S(1)}_{\tau\lambda},\ldots,\texttt{S(LS}_\lambda\texttt{)}_{\tau\lambda}$

Bedeutung der Variablen:

CCNAME,...	Beschreibung des erzeugenden Codes (siehe [1b]).
CJNAME,...	Beschreibung des erzeugenden Laufs (siehe [1b]).
CINAME,...	Beschreibung des Erstlaufs bei Restarts (siehe [1b]).
CIIDE1	Name des gerechneten Problems (siehe [1b]).
CIIDE2	Name des gerechneten Problems (siehe [1b]).
ISYS	Indikator für das (räumliche) Koordinatensystem des darzustellenden Gebiets [1b].
IZ	Indikator für die (räumliche und zeitliche) Dimension des darzst. Gebiets (siehe 2.1).
NR	Räumliche plus zeitliche Dimension des darzustellenden Gebiets.
NS	Räumliche plus zeitliche Dimension des Darstellungsraums.
IR, JR, KR	Anzahl der Maschenhüllen- oder Zeitebenen in den NR Gebietsrichtungen; 0 bei Punktmengen.
IRJRKR	Bei einparametrischen Koordinaten: 0; sonst: Anzahl IR[*JR[*KR]] aller Maschen im Gebiet bzw. Anzahl aller Punkte bei Punktmengen.
LN	IR[*JR[*KR]].
LC	(IR-1)[*(JR-1)[*(KR-1)]].
CI, CJ, CK	Label der NS Koordinatenrichtungen.
XR, YR, ZR	Koordinatenwerte der Maschenhüllen oder Gitterpunkte in den NS Koordinatenrichtungen bzw. aller Maschen im Gebiet.
KT	Wenn IZ < 0: Anzahl der Problemzeitpunkte; sonst 0.
KS	Anzahl der Komponenten.
KL	Anzahl weiterer Koordinatenanordnungen.
TK	Wenn IZ < 0: Werte der Problemzeitpunkte.
CS1,CS2	Labels (1. und 2. Teil) zur Bezeichnung der Komponenten.
KK	0, wenn die Komponente einen Skalar darstellt; n, wenn die Komponente die $n$te Vektorkomponente eines Vektors ist.
LL	$= 0$, wenn die Komponente Lokation 99 hat (LS=LN, IS=IR, JS=JR, KS=KR); < 0, wenn die Komponente cell-based ist (LS=LC, IS=IR-1, JS=JR-1, KS=KR-1); $= \alpha > 0$, wenn die Koordinaten der Komponenten explizit durch die Anordnung α gegeben sind (LS=LA$_\alpha$, IS=IA$_\alpha$, JS=JA$_\alpha$, KS=KA$_\alpha$).
	Für die Lokationsanordnung α:
IA, JA, KA	Anzahl der Lokationsebenen in den NR Gebietsrichtungen.
IAJAKA	Bei einparametrischen Koordinaten: 0; sonst: Anzahl IA[*JA[*KA]] aller Lokationen im Gebiet.
LA	IA[*JA[*KA]].
XA, YA, ZA	Koordinatenwerte der Lokationen in den NS Koordinatenrichtungen bzw. aller Lokationen im Gebiet.
S	Werte der Komponenten.
LS	=IS[*JS[*KS]]: Anzahl der Werte der Komponenten.

Anmerkung: Als „Gebiet“ bezeichnen wir das in VISAPTER ausgewählte Netz/Teilgebiet (oder eine Punktmenge) mit |IZ| räumlichen Dimensionen, plus der zeitlichen Dimension, falls IZ ≥ 0 ist, d.h. insgesamt NR Dimensionen. Unter Umständen kann die (räumliche plus zeitliche) Dimension des Darstellungsraums NS höher sein als die des Gebiets NR (nicht für Gsharp). Die Eingabezeilen 6 bis 10 beschreiben u.a. die Gebietsgeometrie. Zusätzliche Angaben von Lokationsanordnungen sind dann nötig, wenn die Komponenten nicht den hierdurch definierten Maschen (als ganzen) oder Gitterpunkten des Netzes zugeordnet werden können, sondern auf anderen Orten im Netz liegen (z.B. in den Maschenmitten oder auf den Maschenhüllen).

6.3.3 Gsharp-Aufruf

Die von VISAPTER für Gsharp erstellte Datei enthält alle zur Visualisierung eventuell benötigten Informationen in leicht lesbarer Form. In Ergänzung zu diesem Dateiformat werden dazu kompatible Gsharp-Script-Language-Files zur Verfügung gestellt, die beim Aufruf des Visualisierungssystems als Parameter übergeben werden können, und unter Gsharp die Datei interpretieren, die Daten im Gsharp-Datapool ablegen und anschließend den Aufbau einer dem Fall angemessenen Standarddarstellung steuern.

Falls die Datei mit dem Script-Language-File eingelesen wird, stehen die Daten danach als Gsharp-Datasets im Gsharp-Datapool unter den folgenden Namen und Attributen:

Daten (siehe 3.3.2)	Gsharp-Dataset	Datentyp	Datenstruktur
CCNAME,...	g1	TEXT	scalar
CJNAME,...	g2	TEXT	scalar
CINAME,...	g3	TEXT	scalar
CIIDE1	p1	TEXT	scalar
CIIDE2	p2	TEXT	scalar
ISYS	geo	REAL	scalar
IZ	iz	REAL	scalar
NR	dim	REAL	scalar
NS	dims	REAL	scalar
IR, JR, KR	ir, jr, kr	REAL	scalar
IRJRKR	ijkr	REAL	scalar
LN, LC	ln, lc	REAL	scalar
CI, CJ, CK	cx, cy, cz	TEXT	scalar
XR, YR, ZR	xr, yr, zr	REAL	column bzw. grid oder block
KT, KS, KA	n, m, a	REAL	scalar
TK	t	REAL	column
CS1//CS2	c	TEXT	column
CS1//CS2//'at t='//TK	ct1, ct2, ...	TEXT	scalar
KK	vcomp	REAL	column
LL	loc	REAL	column
IA, JA, KA	ia1, ja1, ka1, ...	REAL	scalar
IAJAKA	ijka1, ...	REAL	scalar
LA	la1, ...	REAL	scalar
XA, YA, ZA	xa1, ya1, za1, ...	REAL	column bzw. grid oder block
S	s1, s2, ...	REAL	column, grid oder block
LS	ls1, ls2, ...	REAL	scalar
IS, JS, KS	is1, js1, ks1, ...	REAL	scalar

(Die Gsharp-Datasets s1, s2, ... haben die Struktur „column“, wenn IZ $= -1$ oder IZ $= 0$ ist; „grid“, wenn IZ $= -2$ oder IZ $= 1$ ist; „block“, wenn IZ $= -3$ oder IZ $= 2$ ist.; ebenso xr, yr, zr, falls IRJRKR $\neq 0$ ist, und xa, ya, za, falls IAJAKA $\neq 0$ ist.)

Aus den übergebenen Daten kann Gsharp erkennen, ob eine Darstellung für 1D oder 2D, Skalar- oder Vektorgrößen, node-based oder cell-based, Netze ohne oder mit Löchern, angemessen ist. Der Benutzer muß lediglich noch entscheiden, welche Repräsentation er für zweidimensionale Skalarplots wünscht.

Es genügen deshalb z.Z. die folgenden beiden Gsharp-Script-Language-Files, die sich lediglich in der Repräsentation von 2D-Skalaren unterscheiden (als perspektivische Plots bzw. als Konturplots):

```
/fzk/inr/@sys/VISART/gsharp-slf/gsharp.gsl
/fzk/inr/@sys/VISART/gsharp-slf/gsharpc.gsl
```

Gsharp wird hier mit dem Kommando

`Gsharp -editors -logo` *Script-Language-File*

aufgerufen. Der Script-Language-File bewirkt das automatische Einlesen der von VISAPTER erzeugten Datei unter dem Namen `VINPUT_3.dat` und steuert anschließend eine dem Problem angepaßte grafische Darstellung der als erste eingelesenen Größe `s1`. Anschließend kann der Benutzer die Steuerung von Gsharp zur Manipulation der eingelesenen Daten und Darstellungsparameter selbst übernehmen.

Das Visualisierungssystem Gsharp kann auch über die grafische Benutzeroberfläche [1a] aufgerufen werden. Im Anschluß an das „Ausführen“ des Programms VISAPTER über das VISAPTER-Fenster (Bild 3) der grafischen Benutzeroberfläche wird ein Unterfenster (Bild 5) zum Aufruf von Gsharp angeboten. Alternativ kann das Unterfenster auch durch Betätigen der Taste „Visualisieren“ im VISAPTER-Fenster eröffnet werden, wenn eine Datei `VINPUT_3.dat` bereits vorhanden ist.

Im Gsharp-Unterfenster kann der Benutzer unter den beiden Standard-Script-Language-Files wählen oder auch andere, z.B. selbst erstellte bzw. modifizierte Script-Language-Files spezifizieren. Durch Betätigen der Taste „Visualisierung“ wird Gsharp über das obige Kommando aufgerufen. Nach Beendigung der Gsharp-Sitzung wird in das VISAPTER-Fenster der Benutzeroberfläche zurückgekehrt.

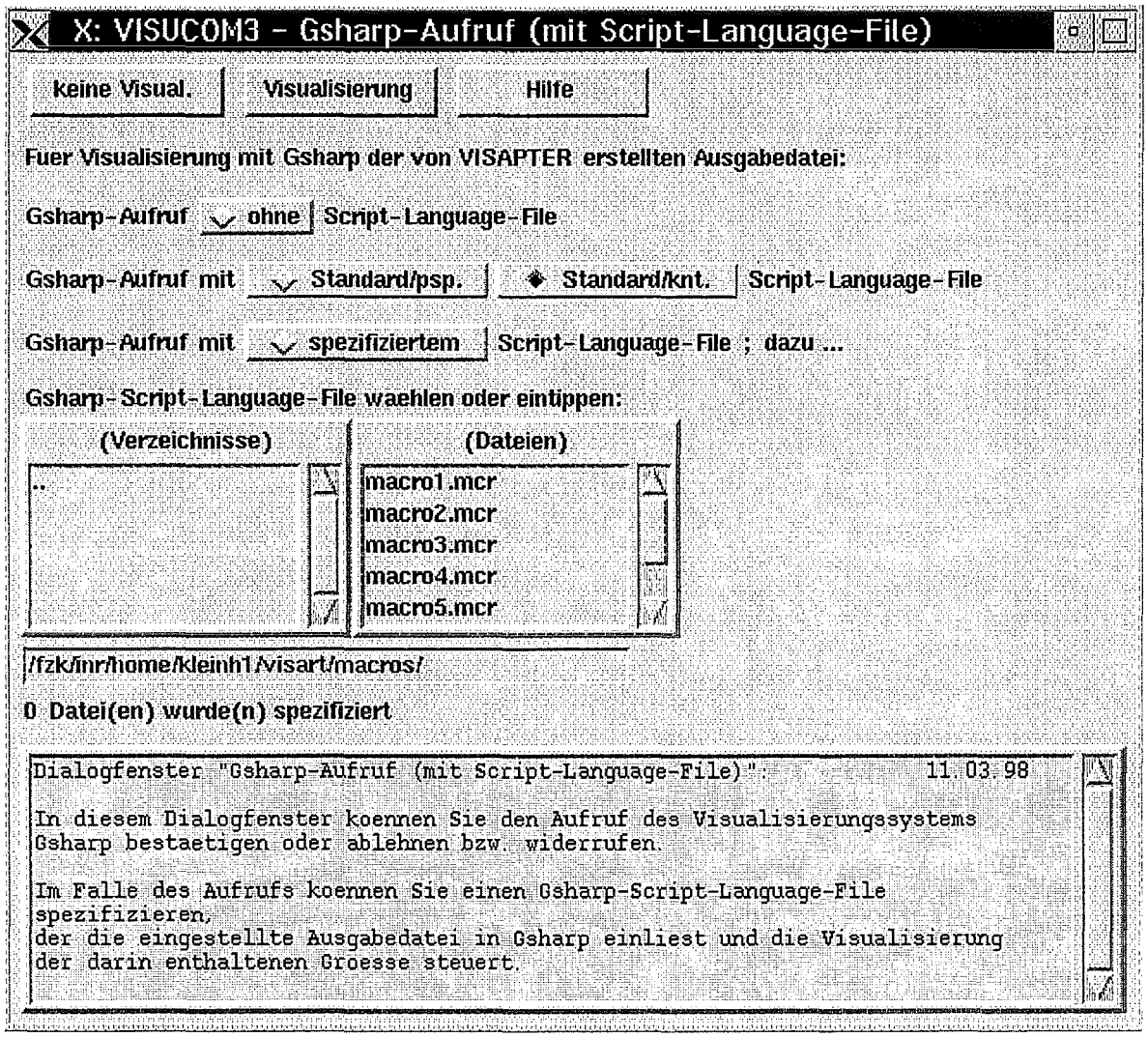


Bild 5: Visualisierungs-Fenster für Gsharp

6.3.4 Beispiele

Bild 5.a zeigt ein Beispiel für die Visualisierung von Daten im VISART-Postprocessor-Format mit Gsharp. Die zugehörige Steuereingabe für das Programm VISAPTER steht in der Liste 5.a.

Das gezeigte PREMIX-Problem aus dem SIMMER-III-Code beruht auf einem zweidimensionalen, regulären, vollen 14×34 -Netz in r, z-Geometrie. Dargestellt wird der Druck, Gruppe `PK`, als Konturplot, zusammen mit der Geschwindigkeit, Gruppe `VEL1`, als Vektorplot, jeweils node-based und an der z-Achse gespiegelt. Das Bild wird mit dem Script-Language-File `gsharpc.gsl` aus der von VISAPTER erzeugten Datei produziert.

Liste 5.a: VISAPTER-Steuereingabe zu Bild 5.a

```
3 -2 -1  0  0  4  4
1  1  1  2
0.98  0.99
15  'PK'  ''  0  -1 -1  -1 -1  -1 -1
15  'VEL1'  ''  0  -1 -1  -1 -1  -1 -1
```

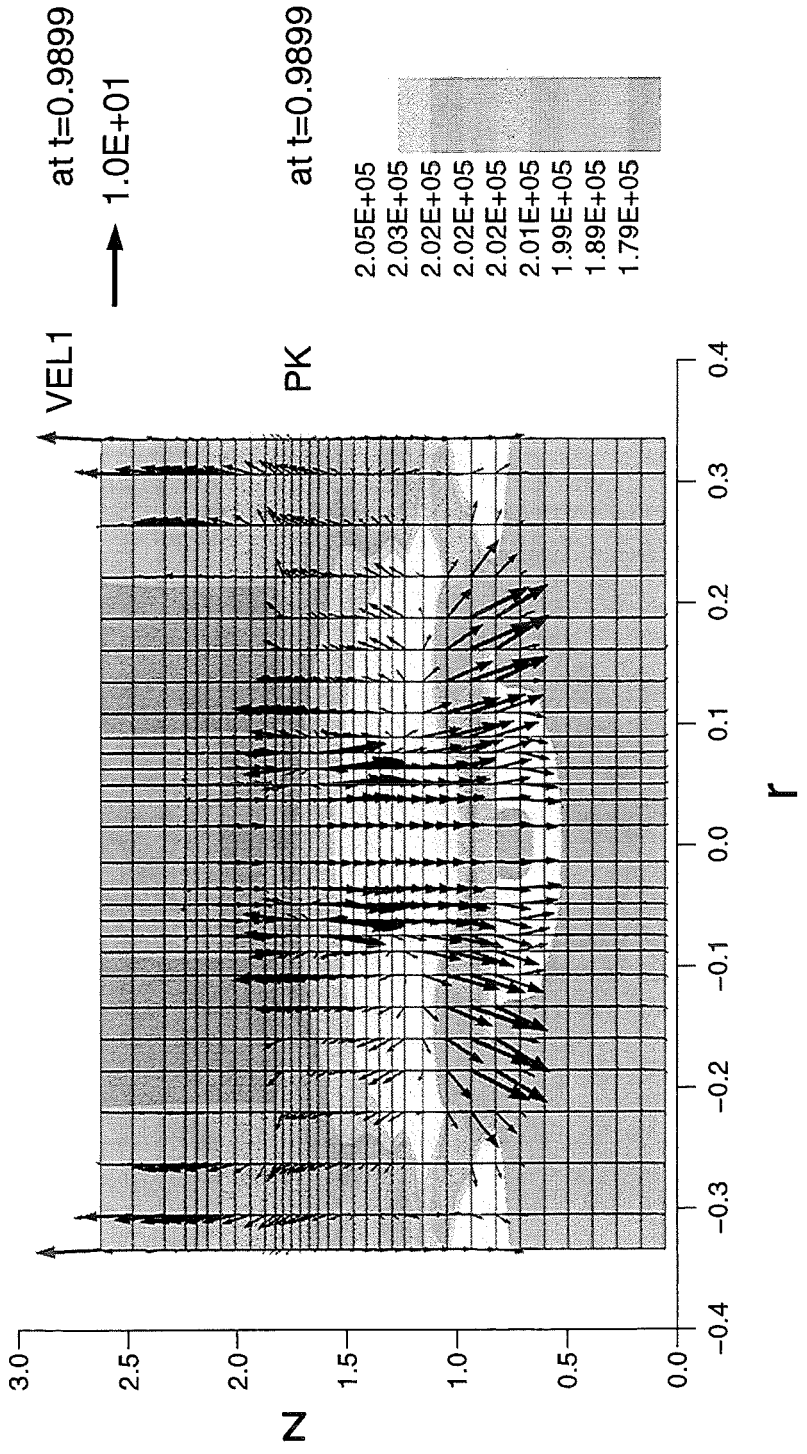


Bild 5.a: PREMIX-Problem aus dem SIMMER-III-Code

6.4 Visualisierung mit Tecplot-7

6.4.1 Besonderheiten

MODE = ±5 wählt die Ausgabe für Tecplot, Version 7 [5a], und zwar mit positivem Vorzeichen eine node-based Darstellung, mit negativem Vorzeichen eine cell-based Darstellung. Cell-based Darstellungen lassen sich zwei- und dreidimensional sinnvoll nur durch Konturplots (dreidimensional evtl. mit Schnitten) repräsentieren, wobei in Tecplot „Corner Flooding" verwendet wird. In der Tecplot-Eingabedatei werden dazu die Werte dem „Principal Data Point" der zugehörigen Masche zugeordnet, d.h. dem Eckpunkt der Masche mit dem kleinsten i-, j- und k-Index[1].

Falls Teilnetze in der VISART-Datei definiert sind, bereitet VISAPTER deren Darstellung als „Zones" in Tecplot vor (das ist nicht möglich zusammen mit Flächenscharen!). Falls Flächenscharen bei der VISAPTER-Ausführung für die Darstellung vorbereitet werden sollen, bereitet VISAPTER deren Darstellung als „Zones" in Tecplot vor (das ist nicht möglich zusammen mit Teilnetzen!).

Für schwach defektive Netze, definiert durch Maschenbelegungsgruppen (siehe 4.1) in der VISART-Datei, konstruiert VISAPTER aus der Maschenbelegung eine „Blanking Variable" für Tecplot zur Ausblendung der „Löcher" (siehe 6.4.2). Stark defektive Netze, definiert durch Gruppen 6/16 in der VISART-Datei, expandiert VISAPTER für Tecplot auf volle Netze mit Löchern und verfährt dann wie für schwach defektive Netze.

IZ darf alle Werte -3, ..., 2 annehmen. Für IZ < 0 ist auch mehr als ein Problemzeitpunkt zulässig.

Die zugrunde liegenden Netze/Teilgebiete können kartesisch, krummlinig orthogonal oder irregulär sein. Bei krummlinigen Koordinatensystemen müssen Vektorkomponenten und Koordinaten auf kartesische Koordinaten umgerechnet werden (IKRUM = 1 oder IKRUM = 2).

Bei vektoriellen Größen können nur die im Teilgebiet liegenden Vektorkomponenten dargestellt werden.

Es können mehrere Größen, auch skalar und vektoriell gemischt, über dem gleichen Netz/Teilgebiet, nicht unbedingt mit gleichen Lokationen und daher nicht unbedingt mit gleicher Anzahl von Koordinatenwerten, ausgewählt werden. Es sind Größen des Typs REAL und INTEGER, auch gemischt, erlaubt (letztere werden auf erstere umdefiniert).

[1] Diese Definition führt bei 3D-Netzen zu Mehrdeutigkeiten und deshalb zu teilweise falschen Darstellungen der Maschenhüllen. Auch verlangt die Definition bei 3D-Netzen eine Zuweisung von Werten an die Mascheneckpunkte, die auf der rechten/hinteren/oberen Netzhülle liegen.

6.4.2 Aufbau der Tecplot-Datei

Tecplot erwartet die Eingabedaten in einem festen Format, das aber sehr flexibel ist. Ob die Eingabedatei als „ASCII-Data-File“ (Default-Suffix `.dat`) oder als „Binary-Data-File“ (Default-Suffix `.plt`) erstellt wird, wird durch die Auswahl des auszuführenden VISAPTER-Binärprogramms (siehe 5.2) festgelegt. ASCII-Dateien werden innerhalb von Tecplot nach dem Einlesen in binäre Form übersetzt, was zeitaufwendig sein kann, und sie beanspruchen mehr Platz als Binärdateien. Andererseits wird die Erstellung von Binary-Data-Files von den bisherigen Tecplot-Releases nur halbherzig unterstützt und empfohlen. Hier wird der Aufbau der ASCII-Data-Files beschrieben; auf die Binary-Data-Files werden die gleichen Daten geschrieben, wenn auch in anderer Anordnung und über die von Tecplot bereitgestellten Funktionen.

Ein von VISAPTER erstellter ASCII-Data-File ist aus File Header, Text Record, Geometry Record und Ordered Zone Records, in einer der unten beschriebenen Anordnungen, aufgebaut [5a]. Die Records werden mit Hilfe von Schlüsselworten und Schlüsselparametern gegliedert (die Aufteilung auf Zeilen ist beliebig). Letztere werden in den folgenden Beschreibungen in Schreibmaschinenschrift angegeben, während Variable kursiv geschrieben werden.

Anordnung A

Diese Anordnung wird für effektive Dimensionen größer als 1 benützt (IZ < -1 oder IZ > 0), sowie für effektive und geometrische Dimension gleich 1 (IZ $= -1$), falls cell-based Darstellung oder mehr als 1 Teilnetz oder Darstellung in 3D vorliegt. Die Koordinaten der Lokationen im Netz bzw. Teilnetz und (bei IZ > 0) des Teilabschnitts der Problemzeit müssen für alle Komponenten gleich sein.

Es können beliebig viele Komponenten vorkommen. Als Komponente zählt neben den aus den spezifizierten Größen abgeleiteten (siehe 4.2) auch die „Value-Blanking-Variable", die Tecplot mitteilt, welche Maschen des Netzes leer sind. Sie wird von VISAPTER, im Falle von Netzen mit „Löchern", als erste Komponente, mit dem Namen BLANK, ausgeschrieben, wobei Werte von 1. die zugeordnete Masche als belegt definieren, Werte von 0. als nicht belegt (Loch). In der folgenden Tabelle sind die Positionen, die nur für Netze mit Löchern benötigt werden, in eckigen Klammern eingeschlossen.

Wenn Größen über den geometrischen Dimensionen 2 und 3 (IZ $= -2$ oder $= -3$) für mehrere Problemzeitpunkte ausgegeben werden, so werden die zugeordneten Komponenten entsprechend mehrfach geschrieben, wobei sie sich durch die Zeitangabe in den zugeordneten Labels unterscheiden lassen. Bei mehr als 10 Problemzeitpunkten werden nur noch die Zeitangaben als Labels geschrieben (und zwar in kleinster Schrift), da angenommen wird, daß es sich um Bilder zur Verfilmung handelt (siehe 6.4.3).

1. TITLE = "*Name des Problems*"
2. VARIABLES = "*vname*$_1$" ... "*vname*$_d$" ["BLANK"]
   "*vname*$_{d+1}$" ... "*vname*$_{d+n}$"

Für jeden Problemzeitpunkt $\tau = 1, \ldots, T$ die folgenden Records 3:

3. TEXT T = "*tname*$_1$" X = *xorigin* Y = *yorigin* C = *color* H = *height*
   ...
   TEXT T = "*tname*$_n$" X = *xorigin* Y = *yorigin* C = *color* H = *height*

Für jedes Teilnetz oder für jede Fläche einer Schar $\mu = 1, \ldots$ die folgenden Records 4, 5, 6:

4. ZONE T = "μ" I = $\bar{i}_1$ [J = $\bar{i}_2$ [K = $\bar{i}_3$]] F = BLOCK
5. *1. Koordinate* (über alle $\bar{i}_1 \times \bar{i}_2[\times\bar{i}_3]$ Lokationen des μ-ten Teilnetzes/Fläche)
   ...
   d-te Koordinate (über alle $\bar{i}_1 \times \bar{i}_2[\times\bar{i}_3]$ Lokationen des μ-ten Teilnetzes/Fläche)
   [*Value-Blanking-Werte*] (über alle $\bar{i}_1 \times \bar{i}_2[\times\bar{i}_3]$ Lokationen des μ-ten Teilnetzes/Fläche)

Für jeden Problemzeitpunkt $\tau = 1, \ldots, T$ die folgenden Records 6:

6. *1. Komponente*$^\tau$ (über alle $\bar{i}_1 \times \bar{i}_2[\times\bar{i}_3]$ Lokationen des μ-ten Teilnetzes/Fläche)
   ...
   n-te Komponente$^\tau$ (über alle $\bar{i}_1 \times \bar{i}_2[\times\bar{i}_3]$ Lokationen des μ-ten Teilnetzes/Fläche)

T ist die Anzahl der Problemzeitpunkte, falls IZ < 0 ist; andernfalls 1;
d ist die effektive Dimension der Darstellungskoordinaten;
n ist die Anzahl der Komponenten;
vname$_{1\ldots d}$ sind die Labels zur Achsenbeschriftung (z.B. x, y, z, r, ..., t);
vname$_{d+1\ldots d+n}$ sind Labels, die hier nicht weiter benutzt werden;
tname$_{1\ldots n}$ sind die Labels zur Bezeichnung der Komponenten (gleich den VISART-Identifikationen der Größen, bei IZ < 0 ergänzt um "at t=t_τ");
xorigin, yorigin, color, hight sind numerische Werte für die Bildkoordinaten usw. der Labels;
$\bar{i}_{1\ldots d}$ sind die Anzahl der Lokationen des μ-ten Teilnetzes bzw. der μ-ten Fläche einer Flächenschar in den Koordinatenrichtungen bzw. die Anzahl der Problemzeitpunkte (bei IZ > 0).

Anordnung B

Diese Anordnung wird für die geometrische Dimension gleich 0 benützt (IZ = 0), sowie für effektive und geometrische Dimension gleich 1 (IZ = −1), falls node-based Darstellung, nur 1 Teilnetz und keine Darstellung in 3D vorliegt. Die ausgewählten Koordinaten der Lokationen des Netzes oder (bei IZ = 0) der Zeitachse müssen nicht für alle Komponenten gleich sein.

1. TITLE = "*Name des Problems*"
2. VARIABLES = "*vname*" "*Q*"

Für jede Komponente $\nu = 1, \ldots, n$ die folgenden Records 4, 5, 6:

4. ZONE T = "*zname*$_\nu$" I = $\bar{\ell}_\nu$ F = BLOCK
5. *Koordinate*$_\nu$ (über alle $\bar{\ell}_\nu$ Lokationen der ν-ten Komponente)
6. *Komponente*$_\nu$ (über alle $\bar{\ell}_\nu$ Lokationen der ν-ten Komponente)

n ist die Anzahl der Komponenten;
vname ist der Label zur Ordinatenbeschriftung (z.B. x, y, z, r, ..., t);
Q ist ein Label, der hier nicht weiter benutzt wird;
zname$_\nu$ sind die Labels zur Bezeichnung der Komponenten (gleich den VISART-Identifikationen der Größen, bei IZ < 0 ergänzt um "at t=t");
$\bar{\ell}_n$ ist die Anzahl der Lokationen in der ausgewählten Koordinatenrichtung bzw. die Anzahl der Problemzeitpunkte der n-ten Komponente.

Anordnung C

Diese Anordnung wird nur für Scatter-Plots benützt, wobei nur `IZ` < 0 möglich ist. Die Anzahl der Komponenten muß für alle Problemzeitpunkte und alle Partikelsorten gleich sein.

1. `TITLE = "`*Name des Problems*`"`

2. `VARIABLES = "`$vname_1$`" ... "`$vname_d$`" "`Q_1`" ... "`Q_n`"`

7. `GEOMETRY T = LINE M = GRID C = BLACK`
   lines

lines-mal der Record 7a:

7a. *Anzahl* der Polylinienstützpunkte und deren *Koordinaten* für die Gebietsumrandung

Für jeden Problemzeitpunkt $\tau = 1, \ldots$ und für jede Partikelsorte $\sigma = 1, \ldots$ die folgenden Records 4, 5, 6:

4. `ZONE T = "`$sname_\sigma^\tau$`" I =` $\bar{\ell}_\sigma^\tau$ `F = BLOCK`

5. *1. Koordinate*$_\sigma^\tau$ (über alle $\bar{\ell}_\sigma^\tau$ Partikel der Sorte σ zum Problemzeitpunkt τ)
   ...
   d-te Koordinate$_\sigma^\tau$ (über alle $\bar{\ell}_\sigma^\tau$ Partikel der Sorte σ zum Problemzeitpunkt τ)

Falls $n > 0$, die folgenden Records 6:

6. *1. Komponente*$_\sigma^\tau$ (über alle $\bar{\ell}_\sigma^\tau$ Partikel der Sorte σ zum Problemzeitpunkt τ)
   ...
   n-te Komponente$_\sigma^\tau$ (über alle $\bar{\ell}_\sigma^\tau$ Partikel der Sorte σ zum Problemzeitpunkt τ)

d ist die effektive Dimension;
n ist die Anzahl der Komponenten (hier Partikeleigenschaften);
lines $= d \times 2^{d-1}$ ist die Anzahl der Polylinien für die Gebietsumrandung;
$vname_{1\ldots d}$ sind die Labels zur Achsenbeschriftung (z.B. x, y, z, r, ...);
$Q_{1\ldots n}$ sind Labels, die hier nicht weiter benutzt werden;
$sname_\sigma^\tau$ sind die Labels zur Bezeichnung der Partikelsorte (gleich den VISART-Identifikationen der Gruppen 6/16, ergänzt um `"at t=`t_τ`"`);
$\bar{\ell}_\sigma^\tau$ ist die Anzahl der Partikel der Sorte σ zum Problemzeitpunkt τ.

6.4.3 Tecplot-Aufruf

Die von VISAPTER für Tecplot in einem der obigen Formate erstellte Datei kann von Tecplot eingelesen und interpretiert werden. In Ergänzung zu diesen Dateiformaten werden dazu kompatible Tecplot-Macro-Files zur Verfügung gestellt, die beim Aufruf des Visualisierungssystems als Parameter übergeben werden können, und unter Tecplot die Datei einlesen und den Aufbau einer dem Fall angemessenen Standarddarstellung steuern. Leider kann Tecplot aus den Formaten mit den zur Verfügung stehenden Macro-Commands nicht entnehmen, welche Art der Darstellung dem jeweiligen Problem angemessen ist, d.h. ob es sich um Skalar- oder Vektorgrößen, Netze ohne oder mit Löchern, handelt. Der Benutzer muß auch noch entscheiden, welche Repräsentation er für die Größen wünscht.

In Ergänzung zu VISAPTER werden deshalb eine ganze Reihe von Macro-Files für die unterschiedlichen Visualisierungsarten zur Verfügung gestellt. Unter DCE/DFS stehen sie im Verzeichnis

`/fzk/inr/@sys/VISART/tecplot-macros`

unter den Dateinamen
`tpt`$ijk\mu\ell mn$`.mcr`

wo i, j, k, μ, ℓ, m, n für Ziffern stehen, deren Werte die folgenden Parameter verschlüsseln:

- i Bilddimension (1, 2 oder 3)
- j Anzahl der zusammen zu visualisierenden Größen (1, ...)
- k node-based (1) oder cell-based (2) Darstellung
- μ volles Netz (1), Netz mit Löchern (2), oder Negativdarstellung bei Netzen mit Löchern (3); netzunabhängige Darstellung (0)
- ℓ Netz-Skalar (1), Netz-Vektor (2), Netz-Skalar mit -Vektor (3), Partikel-Skalar (4), Partikel-Vektor (5); Integral-Skalar (0)
- m Repräsentation der Werte der darzustellenden Größen (in Abhängigkeit von i und ℓ):
  (1) Funktionsgraph
  (2) Höhenlinienplot für Skalar, Stromlinien für Vektoren, Symboldurchmesser für Partikel
  (3) Konturplot für Skalare, Pfeile für Vektoren, Symbolfarbe bzw. Vektorpfeile für Partikel
  (4) dreidimensionaler Plot für 2D-Skalare, Iso-Flächen für 3D-Skalare
  (5) für 2 Skalare: dreidimensionaler Plot mit farbigen Konturen
  (6) für 1 Skalar und 1 Vektor: Konturplot mit Vektorpfeilen
  (7) für 1 Skalar und 1 Vektor: Konturplot mit Stromlinien
- n Einzelbild (1) oder Bildfolge für die Verfilmung (2)

Beispiel: der Macro-File `tpt2111131.mcr` erzeugt einen Konturplot in node-based Darstellung einer Skalargröße über einem vollen zweidimensionalen Netz bzw. Gebiet.

Tecplot wird hier mit dem Kommando

`tecplot -p` *Macro-File*

aufgerufen. Der Macro-File bewirkt das automatische Einlesen der von VISAPTER erzeugten Datei unter dem Namen `VINPUT_5.dat` und steuert anschließend die Darstellung der Größe(n). [1] Anschließend kann der Benutzer die Steuerung von Tecplot zur Manipulation der eingelesenen Daten und Darstellungsparameter selbst übernehmen.

[1] Die Macros zur Verfilmung sind besonders aufwendig im Hinblick auf die Einblendung der Problemzeit der einzelnen Filmbilder. Aus den z.Z. bis zu 600 möglichen, in kleinster Schrift unter- und nebeneinander stehenden Labels der einzelnen Bilder (siehe 6.4.2) vergrößert das Macro den jeweils zutreffenden Label, setzt ihn an die richtige Bildposition und unterdrückt die anderen Labels.

Das Visualisierungssystem Tecplot kann auch über die grafische Benutzeroberfläche [1a] aufgerufen werden. Im Anschluß an das „Ausführen“ des Programms VISAPTER über das VISAPTER-Fenster (Bild 3) der grafischen Benutzeroberfläche wird ein Unterfenster (Bild 6) zum Aufruf von Tecplot angeboten. Alternativ kann das Unterfenster auch durch Betätigen der Taste „Visualisieren“ im VISAPTER-Fenster eröffnet werden, wenn eine Datei `VINPUT_5.dat` bereits vorhanden ist.

Im Tecplot-Unterfenster kann der Benutzer unter den vorhandenen Macro-Files für die Standarddarstellungen wählen, indem er mit den Menütasten die Kombination der in Frage kommenden oder gewünschten Parameter einstellt (entsprechend den Stellen μ, ℓ, m, n im Dateinamen des gewählten Macro-Files; die Werte der Stellen i, j, k werden aus den Einstellungen im VISAPTER-Fenster übernommen). Der Name des voreingestellten Macro-Files wird hinter den Menütasten angezeigt.[1]

Speziell für die Auswahl von Bildfolgen für die Verfilmung kann über die Taste „Einstellungen“ ein weiteres Unterfenster geöffnet werden, in dem z.B. Anzahl und Werte der Niveaus von Konturplots festgelegt und die gewünschte Filmnorm ausgewählt werden kann. Die hier eingestellten Werte modifizieren den oben gewählten Macro-File an den dafür vorgesehenen Stellen, bevor er an Tecplot übergeben wird.

Anstelle eines Standard-Macro-Files kann der Benutzer auch andere, z.B. selbst erstellte bzw. modifizierte Macro-Files spezifizieren.

Durch Betätigen der Taste „Visualisierung“ wird Tecplot über das obige Kommando aufgerufen. Nach Beendigung der Tecplot-Sitzung wird in das VISAPTER-Fenster der Benutzeroberfläche zurückgekehrt.

[1] Fehlermeldungen über das Nichtvorhandensein des gewählten Macro-Files weisen u.U. auf sinnlose Parameterkombinationen hin, oder einfach auf den Umstand, daß für die (bisher noch nicht verlangte) Parameterkombination noch kein Macro-File existiert, was jederzeit nachgebessert werden kann.

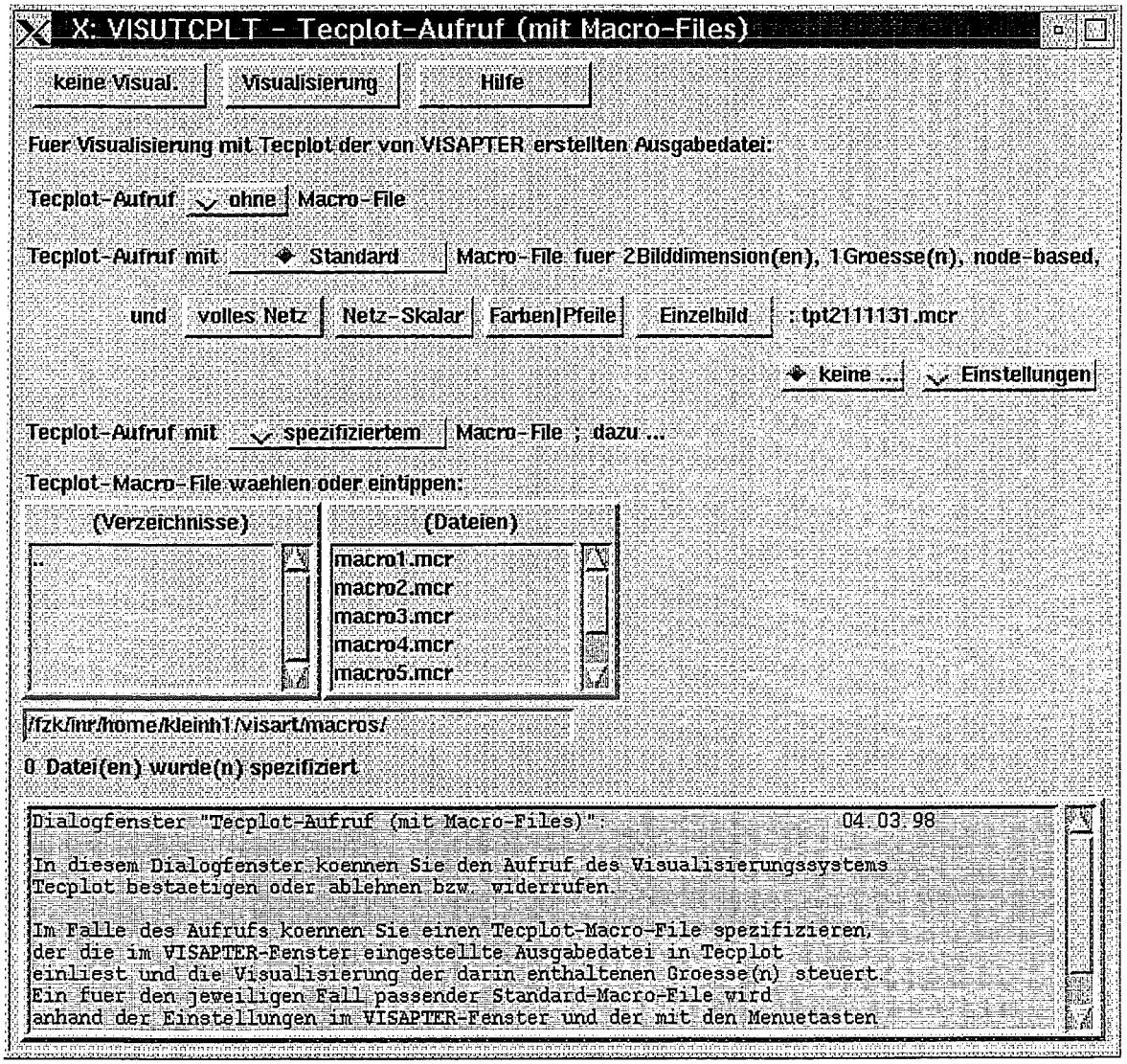


Bild 6: Visualisierungs-Fenster für Tecplot

6.4.4 Beispiele

Bild 6.a, ..., Bild 6.d zeigen Beispiele für die Visualisierung von Daten im VISART-Postprocessor-Format mit Tecplot-7. Die zugehörigen Steuereingaben für das Programm VISAPTER stehen in Liste 6.a, ..., Liste 6.d.

Beispiel 6.a stammt aus dem MATTINA-Code. Das gezeigte FARO-Problem beruht auf einem quasi-zweidimensionalen, regulären, vollen $11 \times 1 \times 45$ -Netz in Zylindergeometrie. Dargestellt wird der Volumenanteil, Gruppe `AL_2`, als perspektivischer Plot, zusammen mit der Temperatur, Gruppe `T_2`, als Konturplot, jeweils node-based und an der z-Achse gespiegelt. Das Bild 6.a wird mit dem Macro-File `tpt2211151.mcr` aus der von VISAPTER mit der Eingabe aus Liste 6.a erzeugten Datei produziert.

Beispiel 6.b stammt aus dem KADI2D-Code. Das gezeigte Dioden-Problem beruht auf einem zweidimensionalen, irregulären, vollen 24×158 -Netz in r, z-Geometrie. Dargestellt wird das Druckfeld, Gruppe `PFELD`, als node-based Konturplot. Das Bild 6.b wird mit dem Macro-File `tpt2111131.mcr` aus der von VISAPTER mit der Eingabe aus Liste 6.b erzeugten Datei produziert.

Beispiel 6.c stammt aus dem GASFLOW-Code. Das gezeigte Tunnel-Problem beruht auf einem dreidimensionalen, regulären $273 \times 22 \times 8$ -Netz mit Löchern in kartesischer Geometrie. Dargestellt wird zum einen das Negativ des belegten Netzes (unter Verwendung einer beliebigen, für die „Löcher" konstanten, Gruppe, hier `r_h2`), als cell-based Konturplot, zum andern die Dichte eines Gasanteils, Gruppe `r_h2`, als node-based Isoflächen, jeweils in einem Teilgebiet des Netzes. (Es handelt sich um die Simulation der Gasausbreitung in einem Tunnel nach einem Tanklastzug-Unfall.) Das Bild 6.c_1 wird mit dem Macro-File `tpt2111131.mcr` aus der von VISAPTER mit der Eingabe aus Liste 6.c_1 erzeugten Datei produziert, das Bild 6.c_2 mit dem Macro-File `tpt2111131.mcr` aus der mit der Eingabe aus Liste 6.c_2 erzeugten Datei.

Beispiel 6.d stammt ebenfalls aus dem GASFLOW-Code. Das gezeigte Problem beruht auf einem dreidimensionalen, regulären $10 \times 23 \times 39$ -Netz mit Löchern in Zylindergeometrie. Außer den durch die „Löcher" modellierten Einbauten gibt es in diesem Beispiel auch Wandstrukturen entlang der Maschenhüllen. Sie werden durch Gruppen `IWALL`, `JWALL`, `KWALL`, Gruppenkennzahl 5, Typ Integer, für die Lokationen 11 bzw. 22 bzw. 44, mit Einsen für undurchlässige Wände an den Maschenhüllen und Nullen andernfalls, definiert. Zur Darstellung der Wände müssen diese Gruppen als „Blanking Variable" für Tecplot herangezogen werden. Im Beispiel werden die Wände auf den Maschenhüllen senkrecht zur ϑ-Koordinate als eine Flächenschar dargestellt[1]. Als darzustellende Variable wurde dazu mit dem Werkzeug VISARTOP [1d] die Gruppe `JHILF` über der Lokation 22, mit konstanten Werten auf jeder Fläche, erzeugt. Das Bild 6.d läßt sich dann aus der von VISAPTER mit der Eingabe aus Liste 6.d erzeugten Datei produzieren.

[1] Über einem zweidimensionalen Netz/Subgebiet lassen sich Wände mit dem Adapterprogramm VISARVOL [1e] zur Visualisierung mit Tecplot-7 aufbereiten.

Liste 6.a: VISAPTER-Steuereingabe zu Bild 6.a

```
5 -2 -1  0  0  4  4
4  1  1  2
1.39  1.41
15  'AL    2'  ''  0  -1 -1  -1 -1  -1 -1
15  'T     2'  ''  0  -1 -1  -1 -1  -1 -1
```

Liste 6.b: VISAPTER-Steuereingabe zu Bild 6.b

```
5 -2  0  0  0  4  4
1  1  99  1
15  'PFELD'  ''  0  -1 -1  -1 -1  -1 -1
```

Liste 6.c_1: VISAPTER-Steuereingabe zu Bild 6.c_1

```
-5 -3  0  0  0  4  4
1  1  0  1
15  'r_h2'  ''  0  80 190  -1 -1  -1 -1
```

Liste 6.c_2: VISAPTER-Steuereingabe zu Bild 6.c_2

```
5 -3  0  0  0  4  4
1  1  99  1
15  'r_h2'  ''  0  80 190  -1 -1  -1 -1
```

Liste 6.d: VISAPTER-Steuereingabe zu Bild 6.d

```
-5 -2  1 -2  0  4  4
1  1  0  2
5  'JWALL'  ''  0  -1 -1  -1 -1  -1 -1
5  'JHILF'  ''  0  -1 -1  -1 -1  -1 -1
```

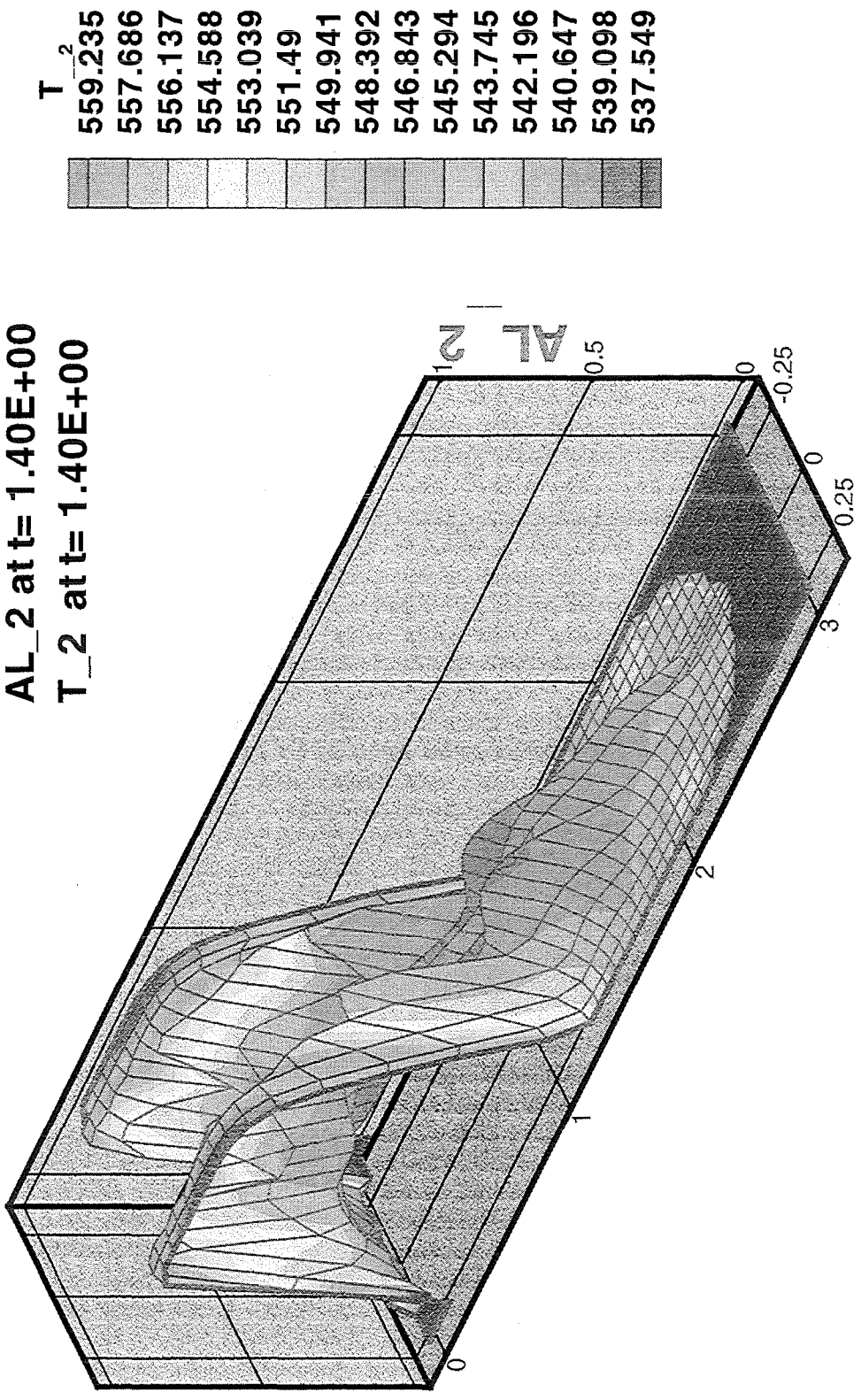


Bild 6.a: FARO-Problem aus dem MATTINA-Code

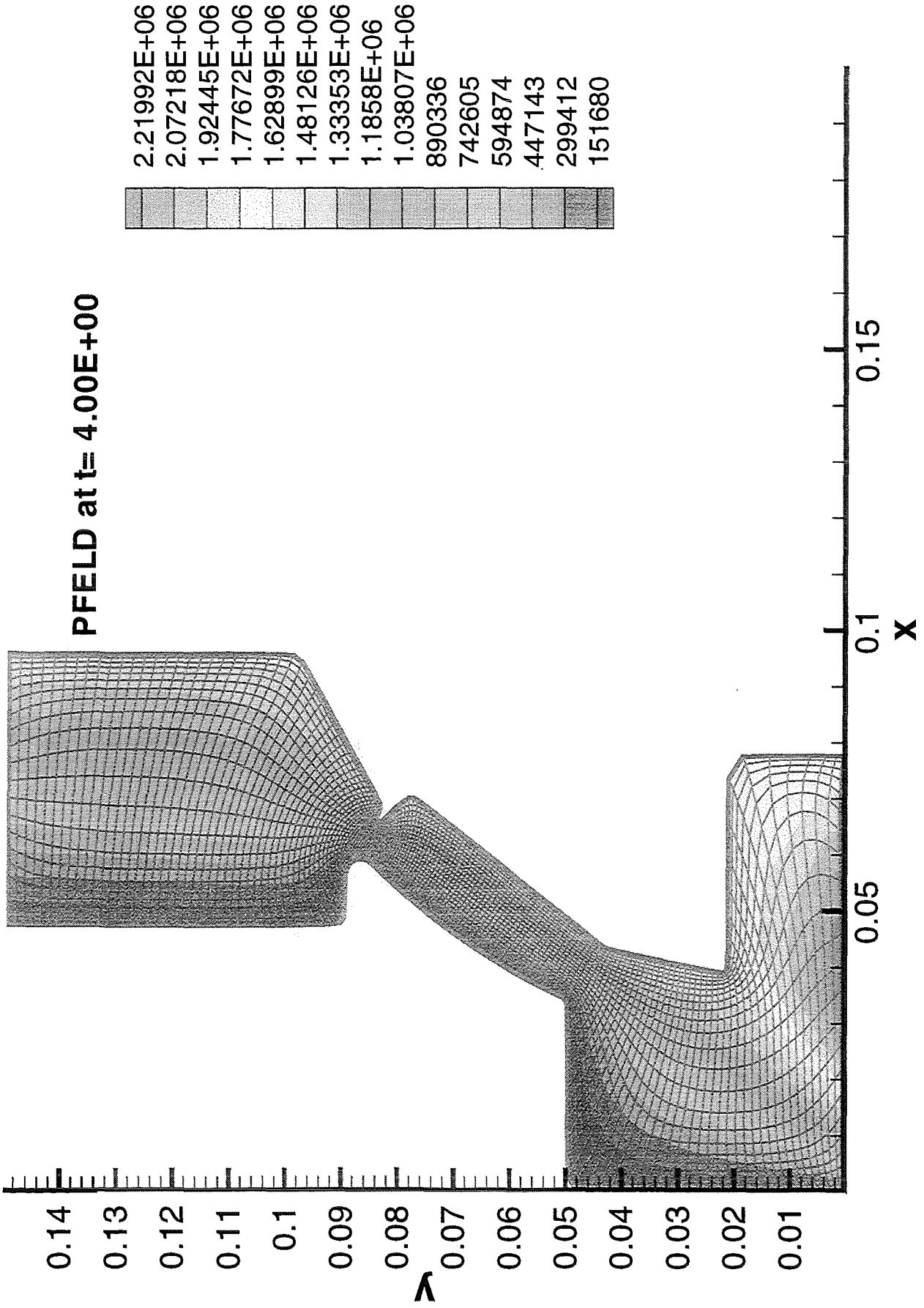


Bild 6.b: Dioden-Problem aus dem KADI2D-Code

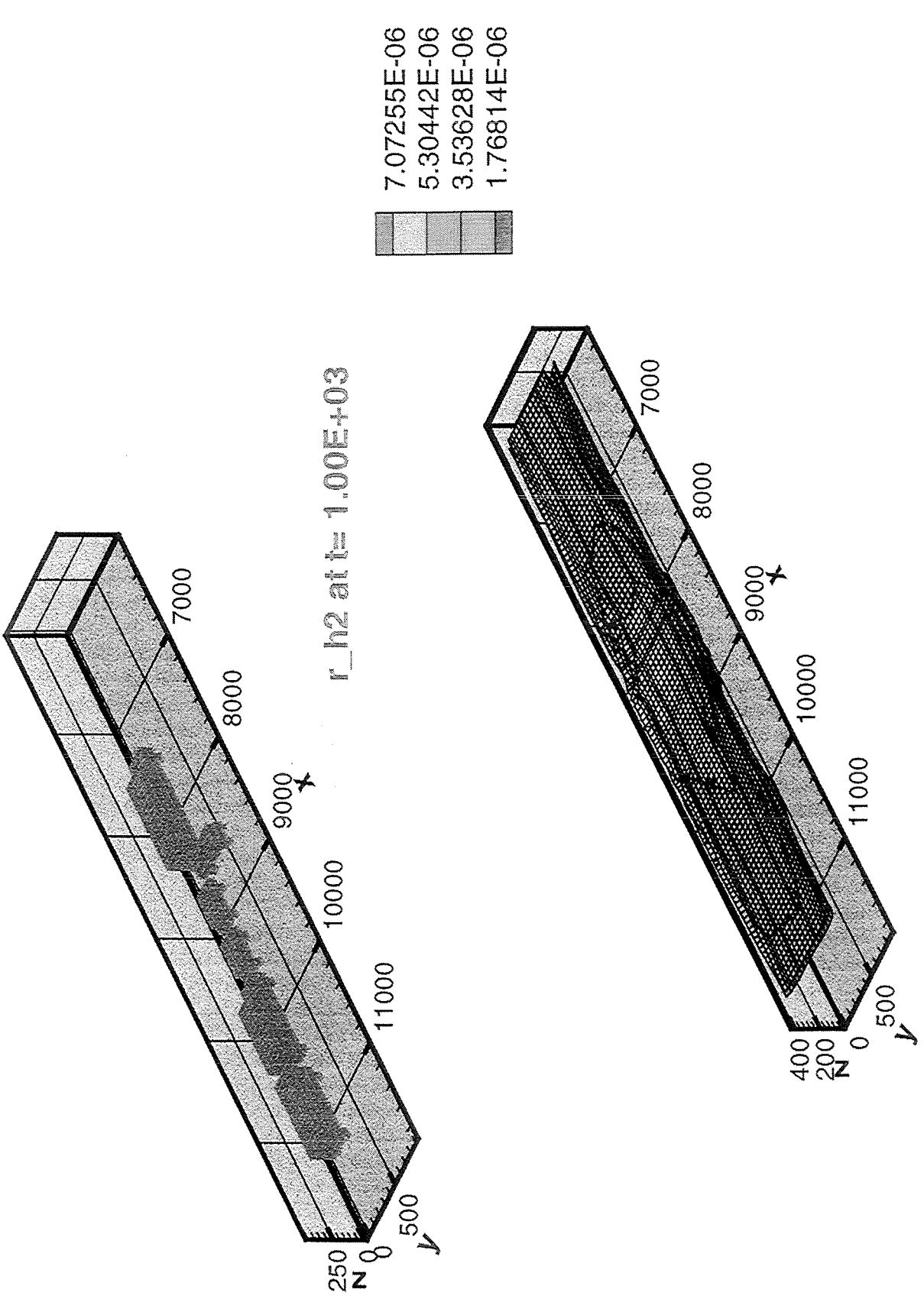


Bild 6.c: Tunnel-Problem aus dem GASFLOW-Code

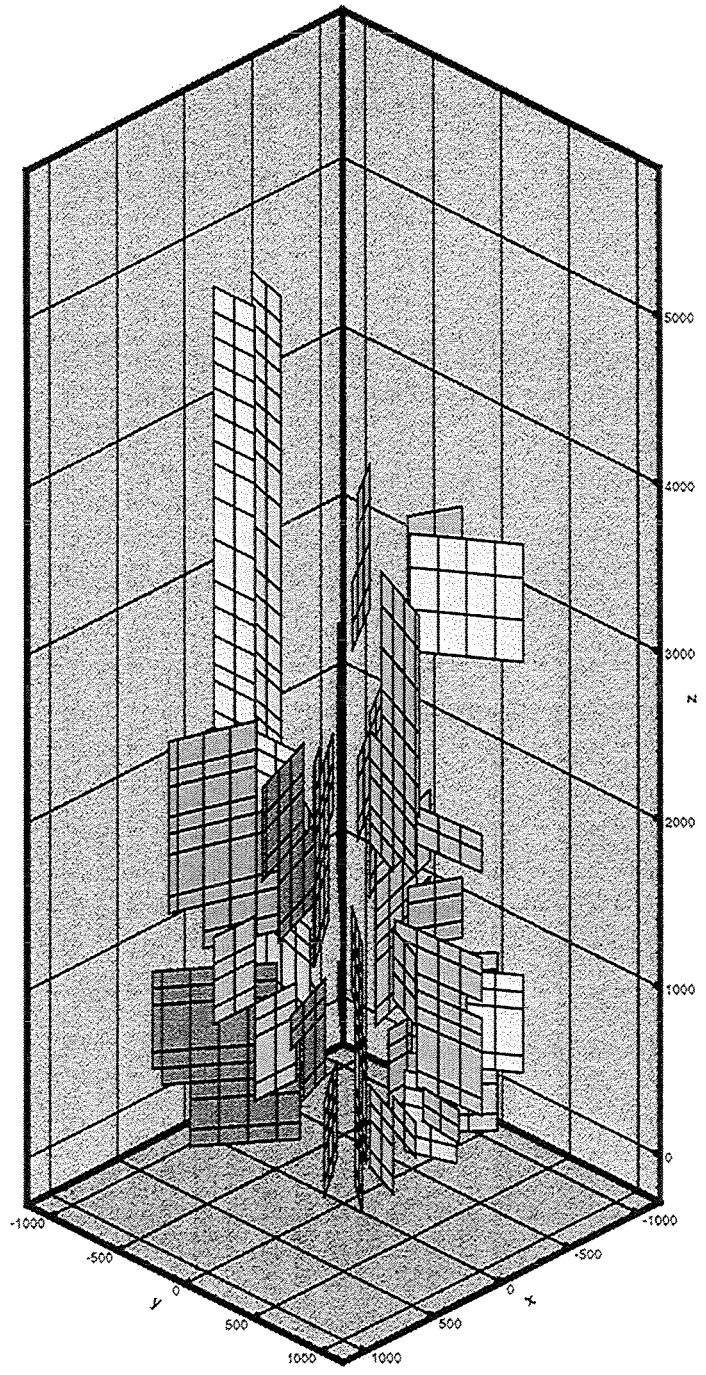


Bild 6.d: Wandstrukturen aus dem GASFLOW-Code

Im folgenden werden noch einige Steuereingabebeispiele für VISAPTER, zur Erzeugung von Funktionsgrafik mit Tecplot, zusammengestellt (ohne Bilder). Sie beziehen sich alle auf den SIMMER-III-Code mit einem zweidimensionalen 3×12 -Netz in r, z-Geometrie.

Die Liste 6.e enthält Steuereingabe für die Darstellung dreier (skalarer) Größen über der Problemzeit, und zwar einer Netzgröße in einer bestimmten Masche des Netzes zu den Problemzeiten der Rumpf-Pakete, der gleichen Größe auch zu *allen* dazwischenliegenden Problemzeitpunkten und der integrierten Größe über dem ganzen Netz zu den Problemzeiten der Rumpf-Pakete.

Die Liste 6.f enthält Steuereingabe für die Darstellung einer (skalaren) Größe über der Problemzeit, und zwar für die Werte der auf einer Traverse liegenden Maschen des Netzes.

Die Liste 6.g enthält Steuereingabe für die Darstellung einer (skalaren) Größe über verschiedenen Traversen durch das Netz zu einem bestimmten Problemzeitpunkt.

Die Liste 6.h enthält Steuereingabe für die Darstellung einer (skalaren) Größe über einer Traverse durch das Netz zu verschiedenen Problemzeitpunkten.

Die Listen 6.i und 6.j enthalten Steuereingabe für die Darstellung einer (skalaren) Größe aus verschiedenen Läufen, und zwar die Werte einer bestimmten Masche über der Problemzeit bzw. die Werte über einer Traverse am Ende der Problemzeit.

Liste 6.e: Darstellung über der Problemzeit

```
5  0  0  0  0  4  4
1  0  0  3
15  'ALPLK 3'  ''  0  1 -1  3 -1  -1 -1
21  'ALPLK 3T'  ''  0  1 -1  3 -1  -1 -1
19  'ALPLK 3'  'INTGRL'  0  -1 -1  -1 -1  -1 -1
```

Liste 6.f: Darstellung über der Problemzeit

```
5  0  0  0  0  4  4
1  0  0  3
15  'ALPLK 3'  ''  0  1 -1  3 -1  -1 -1
15  'ALPLK 3'  ''  0  2 -1  3 -1  -1 -1
15  'ALPLK 3'  ''  0  3 -1  3 -1  -1 -1
```

Liste 6.g: Darstellung über einer Koordinate (mehrere Traversen)

```
5 -1  0  0  0  4  4
1  1  1  3
900.    1000.
15  'ALPLK 3'  ''  0  1 -1  -1 -1  -1 -1
15  'ALPLK 3'  ''  0  2 -1  -1 -1  -1 -1
15  'ALPLK 3'  ''  0  3 -1  -1 -1  -1 -1
```

Liste 6.h: Darstellung über einer Koordinate (mehrere Zeiten)

```
5 -1  0  0  0  4  4
1  9  0  1
15  'ALPLK 3'  ''  0  2 -1  -1 -1  -1 -1
```

Liste 6.i: Darstellung über der Problemzeit (versch. Läufe)

```
5  0  0  0  0  4  4
1  0  0  1
15  'ALPLK 3'  ''  0  1 -1  3 -1  -1 -1
1  0  0  1
15  'ALPLK 3'  ''  0  1 -1  3 -1  -1 -1
```

Liste 6.j: Darstellung über einer Koordinate (versch. Läufe)

```
5 -1  0  0  0  4  4
1  1  99  1
15  'ALPLK 3'  ''  0  1 -1  -1 -1  -1 -1
1  1  99  1
15  'ALPLK 3'  ''  0  1 -1  -1 -1  -1 -1
```

Literatur

[1a] S. Kleinheins:
Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen
Teil 1: Einführung und Überblick
FZKA 5995, 1997

[1b] S. Kleinheins:
Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen
Teil 2: VISART — ein standardisiertes Postprocessor-Dateiformat
FZKA 5996, 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, 1999

[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, 1999

[2] Historian Plus User's Manual
Release 4.3.137
HPCSA, Austin, Texas, August 1991

[3a] AVS User's Guide
Release 3.0, April 1991
Stardent Computer Inc., Concord MA, USA, 1991

[3b] AVS/Express User's Guide
Release 3.0, June 1996
Advanced Visual Systems Inc., Waltham MA, USA, 1992

[3c] AVS/Express Release Notes Addendum
Release 3.4
Advanced Visual Systems Inc., Waltham MA, USA, 1992

[4a] Gsharp User's Guide, Release 2.1
Advanced Visual Systems Inc., Waltham MA, USA, June 1996

[4b] SURFER, Version 4
Golden Software Inc., Golden CO, USA, 1990

[5a] Tecplot User's Manual, Version 7
Amtec Engineering, Inc., Bellevue WA, USA, August 1996

[5b] Tecplot Reference Manual, Version 7
Amtec Engineering, Inc., Bellevue WA, USA