KfK 3911
April 1985

SERVUS
Ein System von Dateien und Prozeduren zur Speicherung und Präsentation von Rechenergebnissen, die Funktionen (einer Variablen) sind

K. Thurnay
Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Neutronenphysik und Reaktortechnik

Projekt Schneller Brüter

KfK 3911

S E R V U S

Ein System von Dateien und Prozeduren zur Speicherung und Präsentation von Rechenergebnissen, die Funktionen (einer Variablen) sind.

K. Thurnay

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

Zusammenfassung:

Das Funktions-Handhabungssystem SERVUS dient dazu , die - bei einer Rechnung anfallenden - Funktionen für spätere Verwendung in geeigneten Dateien zu speichern . Das System ermöglicht es , die gespeicherten Funktionen in einem nachfolgenden Schritt weiterzuverarbeiten oder zu präsentieren , entweder als Zahlenkolonnen , oder als Kurvenscharen oder als Funktionsoberflächen .
Es wird erklärt , wie man eine SERVUS-Datei anlegt,
wie man Funktionen in diese Datei speichert,
wie man Funktionen aus dieser Datei liest,
wie man sich von der Datei einen Katalog beschafft,
wie man eine Kopie der Datei anlegt,
wie man Funktionen einer Datei löscht,
wie man Funktionen der Datei ausdrucken läßt und
wie man Funktionen der Datei als Kurven
oder als 3D-Oberflächen zeichnen läßt .

The Function-Manipulating-System SERVUS.

Abstract:

SERVUS is a service-utility to store functions resulting from a numerical calculation in specially fitted datasets and display them later on. SERVUS enables also the user to manipulate the stored functions, before displaying them. The functions can be presented either as number-columns , or as a family of curves or as surfaces of functions.
The paper describes the steps , needed
for allocating a SERVUS-dataset ,
for storing a function in this set ,
for reading a function from this dataset ,
for cataloguing the functions , stored in the set ,
for copying a SERVUS-dataset ,
for deleting stored functions ,
for printing stored functions and
for drawing two or three-dimensional
pictures of them .

Inhalt:

1. Das Funktions-Handhabungssystem SERVUS

1.1 Das Aufkommen elektronischer Rechner hat dazu geführt , daß man heute auch kompliziert zusammenhängende physikalische Vorgänge nachrechnen und verstehen kann. Die Rechengeschwindigkeit und Speicherkapazität heutiger Rechner erlauben es , diese Vorgänge in den Griff zu bekommen , indem man einfach alle zuständige Bewegungsgleichungen für ein großes Feld von Objektpunkten in einem bestimmten Zeitpunkt löst und dieses Verfahren mit einer kleinen Zeitverschiebung solange fortsetzt , bis der gewünschte Endzeitpunkt erreicht ist.
Leider ist der Ingenieur/Physiker - beim heutigen Stand der FORTRAN--Technik - noch gezwungen , bei der Vorbereitung dieser Rechenverfahren unverhältnismäßig viel Arbeit mit der Lösung von Verwaltungsaufgaben zu verbringen :

- der Speicherplatzbedarf für die beim Rechnen auftretenden Variablen muß im voraus festgelegt werden ,
- die Variablen müssen vor der Rechnung ihre Anfangswerte erhalten,
- die Variablen , deren Entwicklung genauer verfolgt werden soll und die Art , wie ihre Zwischenwerte ausgedruckt werden sollen, müssen im voraus festgelegt werden , usw.

Insbesonders der letzte Punkt - der Ausdruck der Zwischenergebnissen - kostet viel Zeit und ist sehr lästig. Jedesmal, wenn eine neue Variable auftaucht , die man - um die Vorgänge zu verstehen - auch verfolgen möchte , beginnt aufs Neue die Grübelei , auf welche Seite , zusammen mit welchen anderen Größen man sie ausdrucken soll und die Herumplackerei mit den FORMAT-Anweisungen .

Eine weitere nebenwissenschaftliche Arbeit ergibt sich daraus , daß die vom Rechner gelieferten Ergebnisse für den Anwender noch nicht im richtigen , verwertbaren "Zustand" sind. Der Rechner liefert eine große , mehr oder weniger geordnete Zahlenmenge (mitsamt einigen hundert Gramm schwerhandhabbaren Endlospapier) mit der der Anwender in der Regel nicht viel anfangen kann. Um ein Bild von den sich

abspielenden Vorgängen zu gewinnen, muß man diese Zahlengruppen zu Kurven , Kurvenscharen oder zu Funktionsoberflächen umwandeln , ein Vorgang , der sehr viel Arbeit erfordert.

1.2 Das Funktions-Handhabungsystem SERVUS dient dazu , wenigstens einen Teil dieser Verwaltung- und Nachbehandlungs-Arbeit dem Anwender abzunehmen.

Das Grundelement des Systems ist die SERVUS-Datei (s. Kap. 2). Alle Variablen einer Rechnung , die man u. U. zum Verständnis braucht , kann man - anstatt sie im Rechenstrom direkt auszudrucken - als Funktionen in SERVUS-Dateien speichern. Mit Hilfe der SERVUS--Prozeduren kann man dann anschließend die gespeicherte Information präsentieren oder aber auch weiterbehandeln:

Einzelne Funktionen aus einer SERVUS Datei kann man

- löschen ,
- einer laufenden Rechnung zur Verfügung stellen,
- in eine andere SERVUS-Datei kopieren,
- gruppenweise in Standardformat ausdrucken,
- gruppenweise als Kurvenscharen oder als Oberflächen darstellen .

SERVUS-Dateien sind besonders handlich bei den Fortsetzungs--Rechnungen: am Ende einer Fortsetzung werden alle Variablen , die für die Festlegung des System-Zustandes notwendig sind, in eine SERVUS-Datei gespeichert . Die nächste Fortsetzung holt dann diese Funktionen aus der Datei und setzt sie als Anfangswerte der entsprechenden Variablen ein , usw.

1.3 Was braucht ein Benutzer, um mit dem SERVUS-System arbeiten zu können ?

Die zu speichernden Funktionen werden mittels FORTRAN-Subroutinen (s. Kap. 5) in die SERVUS-Dateien eingetragen . Die Handhabung der Dateien erfolgt mit TSO-Kommandoprozeduren und/oder FORTRAN77--Programmen . Der Benutzer braucht somit keine neue Sprachen zu lernen .
Anderseits muß der Benutzer dem SERVUS-System einige Hilfs-Dateien bereitstellen. Neben der SERVUS-Dateien (wie man diese anlegt , s. Kap. 3) braucht man folgende Datenträger:

- Eine gegliederte Datei (partitioned Dataset , s. /1/) tso000.MEMORY zur Aufnahme wichtiger Prozedur-Daten und -Parameter ,

- Eine oder mehrere gegliederte Dateien tso000.buch1 , tso000.buch2 , ... zur Speicherung der gefertigten Bilder.

tso000.MEMORY ist eine standard TSO-Datei (FB / 3120 / 80) . Sie muß mindestens folgende Glieder (Members) enthalten :

DATAS , FIGUR	enthalten die jeweils letzten Parameter-Werte der Kommandoprozeduren DATAS , FIGUR oder DDFIG ,
LAFIGU , LADDFI	enthalten Bildrahmendaten zu den Prozeduren FIGUR bzw. DDFIG ,
ABB1 , ABB2 , ABB3 und ABB4	können den Prozeduren FIGUR , FIGURB , DDFIG oder DDFIGB vorgefertigte Bildunterschriften zuführen .

Es empfiehlt sich , diese Glieder einfach , so wie sie sind aus der Datei TSO105.MEMORY zu übernehmen .

Die Buch-Dateien sind DCB.U-Dateien (Beispiel : 'INR105.GSBOOK'). Jedes gespeichertes Bild kommt in ein eigenes Glied dieser Datei. Für

Näheres über die Buch-Dateien s. das GS-Handbuch /2/.

Schließlich braucht man noch die TSO-Kommando-Prozeduren und die Stapel-(CNTL-)-Prozeduren zur Handhabung der Dateien und der Funktionen.

Die Kommando-Prozeduren	IDA	(Kap. 3)	,
	ENDE	(Kap. 4)	,
	DATAS	(Kap. 6)	,
	FIGUR	(Kap. 9)	und
	DDFIG	(Kap. 10)	

kann man aus der Datei TSO105.SERVUS.CLIST übernehmen , indem man die entsprechende Glieder kopiert (s. auch Kap. 11).

Die Stapel-Prozeduren	KOPIERT	(Kap. 6)	,
	LOESCHT	(Kap. 7)	,
	DRUCKT	(Kap. 8)	,
	FIGURB	(Kap. 9)	und
	DDFIGB	(Kap. 10)	

sind in der Datei TSO105.SERVUS.CNTL . Die Jobkarten in den kopierten Prozeduren muß der Benutzer mit den eigenen Daten versehen.

Bemerkung zur Schreibweise der Prozedur-Parameter :
in diesem Bericht werden Namen - die der Benutzer in einer Prozedur selber einsetzen/abändern muß - kleingeschrieben , die festgelegten Namen erscheinen dagegen in Blockschrift. Zum Beispiel wird , bei tsoOOO.MEMORY vom Benutzer erwartet , daß er hier seine eigene Benutzernummer , z.B. TSO999 einsetzt : TSO999.MEMORY .

2. Über SERVUS - Dateien.

2.1 Eine SERVUS-Datei dient dazu , Funktionen F(X) - die bei einer laufenden Rechnung anfallen - für eine spätere Verwendung zu registrieren . Sie ist vom Typ "DATA" und hat den konventionellen Namen proj.nameda.DATA (z. B. INR105.TEST.DATA).

Eine SERVUS-Datei besteht

- aus einer Kopfzeile,
- aus 0 bis mehreren Funktions-Aufzeichnungen
- und aus einer Endzeile .

Die Kopfzeile enthält 4 CHARACTER*8 Wörter , die für die Kennzeichnung und automatische Bearbeitung der Datei gebraucht werden .

Kopfzeile : { NAMEDA NORM NATAL ACTUAL } .

NAMEDA ist der Name der Datei (" Library " im SPF) .
NORM bezeichnet die Schreib-Norm der Datei . NORM legt fest , in welcher Weise die Funktions-Daten in der Datei aufgeschrieben sind . Diese Normen werden im Teil 2.3 beschrieben .
NATAL gibt das Anlege-Datum der Datei an und
ACTUAL enthält den Tag , an dem die Datei zum letzten Mal vollständig neu beschrieben wurde .

Eine Funktions-Aufzeichnung besteht aus einer Namenszeile und aus einer oder zwei Datenzeilen . Die Namenszeile enthält Kennungs-Daten , die Anzahl der Elemente und Zeichen-Hilfen für die Funktion , die Datenzeile die Ordinaten der Funktion . Falls es zwei Datenzeilen gibt , sind die Ordinaten in der zweiten Zeile , in der ersten sind dann die Abszissen der Funktion .

Die Funktions-Aufzeichnungen werden bei der Verarbeitung durchnumeriert . Ein späteres Wiederauffinden einer Funktion erfolgt ausschließlich anhand ihrer Ordnungszahl .

Die Endzeile sieht formal wie die Namenszeile einer Funktions--Aufzeichnung aus, enthält aber an bestimmten Stellen das CHARACTER*8 Wort '**ENDE**'

2.2 E i n s c h r ä n k u n g e n .

Die Verarbeitungs-Prozeduren erfordern gewisse Einschränkungen von den gespeicherten Funktionen: keine darf mehr als 1000 Elemente enthalten .
Funktionen , die mit der DDFIG- oder DDFIGB-Routine als räumliche Oberfläche gezeichnet werden , dürfen höchstens 250 Elemente enthalten . Außerdem muß eine solche Oberfläche mit höchstens 100 Funktionen auskommen . Die Anzahl der Markierungen auf einer Koordinaten-Achse (s. Abb. 8) ist bei dieser Prozedur auf 21 begrenzt .
Mit der FIGUR- bzw. FIGURB-Prozedur lassen sich höchstens 20 Funktionen in einem Bild zusammenlegen .
Eine weitere Einschränkung betrifft die Anordnung der Funktionselementen in der Dateien : um bei den Abbildung keine Fehler zu erzeugen , müssen alle Funktionen mit monoton-wachsenden Abszissen gespeichert sein.

2.3 D i e S c h r e i b - N o r m e n d e r S E R V U S - D a t e i e n .

Im SERVUS-System werden zur Zeit vier Schreib-Normen benutzt : GRA4 , GRA8 , FOL8 und PLOT .

```
++++++++++++++++++++
+ GRA4 = 'GRAPHIC4' +
++++++++++++++++++++
```

Bei der Schreib-Norm GRA4 besteht jede Funktions-Aufzeichnung aus je drei Datengruppen:

1) { KLASSE , NUMMER , NP ,
   XMI , XMA , XED , FMI , FMA , FED,
   NAMX , MASX , NAMF , MASF } ,
2) { X(j) , j=1,NP } ,
3) { F(j) , j=1,NP } .

Die erste Datengruppe dient zur Kennung der Funktion . Sie enthält auch Hilfsgrößen für eine graphische Darstellung .

KLASSE,NUMMER sind zwei INTEGER Zahlen zur Identifikation der Funktion (z.B. Fall-Nr. und Nr. des Zeitschrittes).

NP (INTEGER) registriert die Anzahl der Funktions-Elemente .

Die folgenden 6 Größen sind alle REAL*4 Zahlen .

XMI,XMA sind untere bzw. obere Grenzen des Abszissen-Wertebereiches ,

XED ist die Länge einer "Skala" auf der X-Achse (s. Skizze).

FMI,FMA, FED sind die entsprechenden Größen bei der F-Achse .

Die restlichen 4 Größen in dieser Gruppe sind alle CHARACTER*8 Konstanten . Sie werden eigentlich nur für die Achsenbeschriftung bei einer eventuellen graphischen Darstellung der Funktion benötigt , man kann sie aber auch als zusätzliche Kennungs-Hilfen benutzen .

NAMX ist die Name der X-Achse (z.B 'Temperat') ,

MASX ist die Maß-Einheit (z.B. ' K ') der Achse .

NAMF und

MASF sind die entsprechenden Namen für die F-Achse .

X(j) die zweite Datengruppe enthält die Abszissen und

F(j) die dritte Datengruppe enthält die Ordinaten der Funktion .

Bei der Schreibnorm 'GRAPHIC4' sind sowohl die Abszissen als auch die Ordinaten allesamt REAL*4 Zahlen .

Die Endzeile bei dieser Norm ist die Datengruppe

{ KLASSE,NUMMER,NP,XMI,XMA,XED,FMI,FMA,FED,ENDE,ENDE,ENDE,ENDE }

wobei ENDE = '**ENDE**' ist .

Die Schreib-Norm GRA4 = 'GRAPHIC4' ist unbedingt erforderlich , falls

man die Funktionen graphisch verarbeiten will ; die benötigten Routinen können keine REAL*8 Zahlen verarbeiten .

```
++++++++++++++++++++
+ GRA8 = 'GRAPHIC8' +
++++++++++++++++++++
```

Bei dieser Schreib-Norm sind ebenfalls die gleichen drei Datengruppen je Funktions-Aufzeichnung vorhanden :

1) { $KLASS , $NUMMR , $NP ,
   YMI , YMA , YED , GMI , GMA , GED,
   NAMX , MASX , NAMF , MASF } ,
2) { Y(j) , j=1,NP } ,
3) { G(j) , j=1,NP } .

Die Daten [$KLASS , ... , G(NP)] haben dieselbe Bedeutungen , wie die Daten [KLASSE , ... , F(NP)] bei der Norm GRA4 . Abweichend von der GRA4-Norm sind hier aber sowohl die Zahlen Y , G , YMI , ... , GED , als auch die Nummern $KLASS , $NUMMR , $NP allesamt REAL*8 Zahlen .

Die Endzeile bei der GRA8-Norm ist die Datengruppe

{ $KLASS,$NUMMR,$NP,YMI,YMA,YED,GMI,GMA,GED,ENDE,ENDE,ENDE,ENDE }

mit ENDE = '**ENDE**'.

Der Vorzug dieser Normierung ist , daß man die anfallenden Funktionen ohne Verlust an Genauigkeit speichern kann . GRA8 eignet sich aber nicht zur Bild-Darstellung . Umkopieren ist notwendig!

```
++++++++++++++++++++
+ FOL8 = '*FOLIA*8' +
++++++++++++++++++++
```

Bei der Schreib-Norm FOL8 werden die zu notierenden Funktionen gruppenweise , als "Blätter" aufgezeichnet .
Jedes "Blatt" enthält mehrere Funktions-Aufzeichnungen , wobei aber immer nur die Ordinaten registriert werden . Die Abszissen werden nur einmal - am Anfang jedes "Blattes" - aufgeschrieben und gelten als

gemeinsame X-Koordinaten für alle Funktionen auf diesem "Blatt" .
Die Aufzeichnungen auf einem - auf dem $BL-ten - "Blatt" sehen wie folgt aus:

Als erstes erscheinen die Daten zum Abszissen-Bereich :

```
1,a)      {$KLASS,$NUMMR, $NP, $BL,
          YMI,YMA,YED,NAMX,MASX } ,
1,b)      { Y(j) , j=1,NP } .
```

Dann folgen die Ordinaten-Daten von 1 - $NF Funktionen :

```
2,a)      {$KLASS,$NUMMR,-$BL, 1.D+0,
          GMI,GMA,GED,NAMF,MASF } ,
2,b)      { G(j) , j=1,NP } .
               ............
               ............
               ............
$NF+1,a)  {$KLASS,$NUMMR,-$BL, $NF,
          GMI,GMA,GED,NAMF,MASF } ,
$NF+1,b)  { G(j) , j=1,NP } .
```

Die Bezeichnungsweise der Größen entspricht im Wesentlichen der Bezeichnungsweise der GRA8-Norm . $NP ist die Zahl der Elemente der ersten Funktion . Die Zahl $BL ist die Ordnungszahl des Blattes . In jeder Funktions-Eintragung erscheint auch die Ordnungszahl der Funktion auf diesem Blatt (1.D+0 ,... , $NF) .

Die Endzeile bei der FOL8-Norm lautet:

```
{ $KLASS,$NUMMR,$NP,$NF,GMI,GMA,GED,ENDE,ENDE } .
```

Diese Schreibnorm ist von Vorteil , wenn man beim Aufzeichnen Platz sparen will , setzt aber voraus , daß alle Funktionen eines "Blattes" auf demselben Abszissen-Bereich definiert sind . FOL8 ist auch nicht geeignet für die graphische Darstellung !

++++++++++++++++++++
+ PLOT = 'PLOTEASY' +
++++++++++++++++++++

Dies ist die bekannte PLOTEASY-Aufzeichnungsnorm /3/. Die Datengruppen sind hier:

1) { NP , NAMF , KLASSE , NUMMER } ,
2) { X(j) , j=1,NP } ,
3) { F(j) , j=1,NP } .

Die Abszissen und die Ordinaten sind hier REAL*4 Zahlen , NAMF ist eine CHARACTER*8 Konstante und die restlichen Größen sind INTEGERs . Die Bedeutung der Größen ist dieselbe wie bei der Norm GRA4 .

Die Endzeile bei der PLOT-Norm ist { NP,ENDE,KLASSE,NUMMER } .

PLOT-Dateien eignen sich auch nicht zur direkten Bild-Herstellung !

3. Eine SERVUS-Datei wird angelegt .

Man kann - mit Hilfe der Kommandoprozedur IDA - eine SERVUS-Datei anlegen , oder sie umtaufen/umnormieren . Falls die Datei schon besteht, wird nur die Kopfzeile der Datei neu geschrieben . Falls es sich um eine neue Datei handelt , wird sie erst allokiert und anschließend die Kopfzeile geschrieben . Bei einer Umtaufe/Umnormierung geht der bisherige Inhalt der Datei verloren !

Aufruf : EX 'tso000.SERVUS.CLIST(IDA)' .

Die Prozedur IDA fragt vom Benutzer die erforderlichen Angaben zu der Datei (Project , Name , Platte , Norm) ab und erledigt die Arbeit im Vordergrund . Zum Abschluss wird - als Bestätigung - die neue Kopfzeile der Datei am Bildschirm ausgegeben.

Die Liste der Prozedur IDA ist im Kap. 11 abgedruckt . Dem Benutzer wird empfohlen , diese Datei zu kopieren , wobei die kleingeschriebenen Namen in der Datei mit den eigenen Ausdrücken zu ersetzten sind. In diesem Falle vereinfacht sich der Aufruf zu :

EX SERVUS(IDA) .

Bemerkung:
Beim Neuanlegen einer Datei geht IDA mit dem zu allokierenden Speicherraum recht großzügig um ; jede Datei erhält 10 Zylinder Primärraum. Falls man nicht so viel (oder noch mehr) Platz anlegen will , empfiehlt es sich , die Datei in Handarbeit zu allokieren (SPF-Menü 3.2) und IDA nur die Normierung überlassen.

4. Eine SERVUS-Datei wird mit einer Endzeile versehen .

Jede SERVUS-Datei erhält beim Anlegen eine Endzeile . Es kann aber vorkommen , daß diese zerstört wird , oder verloren geht , z. B. wenn ein Versuch gemacht wird , in eine - schon vollgeschriebene Datei - weitere Funktionen einzutragen.
Eine Datei ohne Endzeile können die Lese- und Schreib-Routinen aber nicht mehr bearbeiten , somit würde der Verlust der Endzeile zum Verlust der ganzen Datei führen.

Die Kommandoprozedur ENDE ermöglicht es , in eine SERVUS-Datei hinter einer beliebigen Funktion eine Endzeile zu setzen. Man kann daher mit dieser Prozedur nicht nur beschädigte Dateien retten , sondern eine lange Liste unerwünschter Funktionen am Ende einer Datei schnell beseitigen.

Aufruf : EX 'tso000.SERVUS.CLIST(ENDE)' .

Die Prozedur arbeitet im Vordergrund. Erst wird vom Benutzer im Kommando-Dialog der Name der betreffenden Datei erfragt , dann setzt ein FORTRAN-Dialog ein zur Ermittlung der zukünftigen Stelle der Endzeile in der Datei . Der Benutzer wird dabei um die Ordnungszahl der letzten , zu erhaltenden Funktion gebeten. Bei der Eingabe dieser Zahl bestehen folgende Alternativen :

"-" Falls man eine negative Zahl eingibt , dann erfolgt die Eintragung der Endzeile an Stelle der zuletzt gelesenen Funktion.

"0" Falls man eine Null eingibt , endet die Prozedur.

"+" Bei der Eingabe einer positiven Zahl wird - normalerweise - die Katalog-Eintragung (Namenszeile der Funktion) zu dieser Ordnungszahl vorgespielt. Falls es aber in der Datei keine Funktion mehr zu dieser Zahl gibt und auch keine Endzeile , dann endet die Prozedur fehlerhaft.

Die Liste der Prozedur ENDE ist ebenfalls in Kap. 11 zu finden.

5. Ein- und Ausgabe der Funktionen bei SERVUS-Dateien .

5.1 Öffnung der Dateien.

Bevor man Funktionen aus einer Datei lesen und/oder welche in eine Datei eintragen kann , muß man diese Dateien in Bereitschaft versetzen : sie müssen symbolische Nummern erhalten , bei einer Schreib--Datei muß der Schreibmodus festgelegt werden , usw. Zu diesem Zweck ist es unerläßlich , daß man vor dem ersten Schreib- oder Lesebefehl die Subroutine SERDIO aufruft.
Mit einem SERDIO-Aufruf kann man gleichzeitig je eine Schreib- und Lesedatei anregen.

Aufruf : CALL SERDIO(IRE,NORMRE,IWR,NORMWR,MODUS,KPRI) .

Folgende Größen sind beim Aufruf anzugeben:

IRE	(INTEGER*4)	Symbolische Nummer der Ein/Ausgabe-Einheit zum Lesen einer Datei . Falls keine Lese-Datei benötigt , so setze man IRE = 0 .
IWR	(INTEGER*4)	Symbolische Nummer zum Schreiben in die Datei . Falls keine Schreib-Datei benötigt , kann man IWR = 0 setzen.
MODUS	(CHARACTER*8)	Schreib-Modus . Er kann folgende Werte haben:
	='**OLD***'	Das Schreiben in die Datei beginnt hinter dem letzten Eintrag , d.h. die alten Daten bleiben erhalten !
	='*NEWPAG*'	Das Schreiben in die Datei beginnt hinter dem letzten Eintrag . Bei einer FOL8-Datei wird außerdem - im Gegensatz zum OLD-Modus - ein neues Blatt aufgeschlagen .
	='**NEW***'	Das Schreiben in die Datei beginnt am Anfang , hinter der Kopfzeile , d.h. die alten Daten werden zerstört!
KPRI	(INTEGER*4)	Maß für die Intensität der Ausgabe :
für	KPRI = 0	wird nichts ausgedruckt ,
für	KPRI = 1	wird beim Schreiben die Namenszeile der notierten Funktion ausgedruckt ,

für KPRI = 2 werden auch die Namenszeilen der gelesenen Funktion ausgedruckt .

KPRI > 2 wird nicht empfohlen . Es werden die Namenszeilen aller Funktionen ausgegeben , die bei der Lese- oder Schreib-Prozedur berührt werden.

Die restlichen Größen werden vom Programm ausgefüllt :

NORMRE (CHARACTER*8) Schreib-Norm der Lese-Datei .
NORMWR (CHARACTER*8) Schreib-Norm der Schreib-Datei .

Anwendungs-Beispiele für diese Routine kann man unter 5.5 finden.

5.2 S c h r e i b e n i n e i n e D a t e i .

Im folgenden werden die zum Schreiben einer Funktion benötigten Routinen zusammengestellt . Die Wahl der Schreib-Routine hängt von der Schreib-Norm der Datei ab .

```
                          ++++++++++++++++++++
Falls die Datei nach der  + GRA4 = 'GRAPHIC4' +      Norm
                          ++++++++++++++++++++
```

beschrieben werden soll , wird die Subroutine SDING4 benutzt .

```
Aufruf : CALL SDING4(KLASSE,NUMMER,NP
                    ,X,XMI,XMA,XED,NAMX,MASX
                    ,F,FMI,FMA,FED,NAMF,MASF
                                       ,MODUS) .
```

Für die folgenden Größen muß man beim Aufruf sinnvolle Werte angeben :

KLASSE (INTEGER) 2 Kenngrößen zur Identifikation
NUMMER (INTEGER) der Funktion .
NP (INTEGER) Anzahl der Funktions-Elemente , NP muß ≤ 1000 sein !
X (REAL*4 Feld) Die Abszissen bzw.
F (REAL*4 Feld) die Ordinaten der Funktion .

MODUS (CHARACTER*8) Schreib-Modus , s. Routine SERDIO .

Die folgenden Größen brauchen beim Aufruf nur formal vorhanden sein:

XMI,XMA	(REAL*4)	Bereichsgrenzen der Abszissen
FMI,FMA	(REAL*4)	bzw. der Ordinaten .
XED	(REAL*4)	Länge einer "Skala" (s. Kap. 2)
FED	(REAL*4)	auf der X- bzw. F-Achse .
NAMX	(CHARACTER*8)	Name für die X- bzw. für
NAMF	(CHARACTER*8)	die F-Achse (z.B. ' ZEIT ').
MASX	(CHARACTER*8)	Einheit für die X- bzw. für
MASF	(CHARACTER*8)	die F-Achse (z.B. ' SEC ').

Bei einer Abbildung der Funktion werden die Koordinatenachsen mit NAMX MASX bzw. NAMF MASF beschriftet .

Falls die Datei nach der

```
++++++++++++++++++++
+ GRA8 = 'GRAPHIC8' +
++++++++++++++++++++
```

Norm

beschrieben werden soll , wird die Subroutine SDING8 verwendet.

```
Aufruf :  CALL SDING8(KLASSE,NUMMER,NP
                     ,Y,YMI,YMA,YED,NAMX,MASX
                     ,G,GMI,GMA,GED,NAMF,MASF
                                    ,MODUS) .
```

Größen , die man beim Aufruf sinnvoll setzen muß :

KLASSE	(INTEGER)	2 Kenngrößen zur Identifikation
NUMMER	(INTEGER)	der Funktion .
NP	(INTEGER)	Anzahl der Funktions-Elemente .
		NP muß ≤ 1000 sein !
Y	(REAL*8 Feld)	Die Abszissen bzw.
G	(REAL*8 Feld)	die Ordinaten der Funktion .
MODUS	(CHARACTER*8)	Schreib-Modus , s. Routine SERDIO .

Größen , die nur formal vorhanden sein müssen:

YMI,YMA	(REAL*8)	Bereichsgrenzen der Abszissen
GMI,GMA	(REAL*8)	bzw. der Ordinaten .

YED	(REAL*8)	Länge einer "Skala"
GED	(REAL*8)	auf der X- bzw. F-Achse .
NAMX	(CHARACTER*8)	Name für die X- bzw. für
NAMF	(CHARACTER*8)	die F-Achse (z.B. ' ZEIT ').
MASX	(CHARACTER*8)	Einheit für die X- bzw. für
MASF	(CHARACTER*8)	die F-Achse (z.B. ' SEC ').

Falls die Datei nach der

```
++++++++++++++++++++
+ FOL8 = '*FOLIA*8' +
++++++++++++++++++++
```

Norm

beschrieben werden soll , ruft man die Subroutine SDINF8 auf .

Aufruf :

```
CALL SDINF8(KLASSE,NUMMER,NP
           ,Y,YMI,YMA,YED,NAMX,MASX
           ,G,GMI,GMA,GED,NAMF,MASF
                          ,MODUS,NEWP) .
```

Größen , die beim Aufruf sinnvolle Werte verlangen :

KLASSE	(INTEGER)	2 Kenngrößen zur Identifikation
NUMMER	(INTEGER)	der Funktion .
NP	(INTEGER)	Anzahl der Funktions-Elemente . NP muß ≤ 1000 sein !
NEWP	(integer)	Für NEWP = 1 wird die Funktion auf ein neues "Blatt" geschrieben , sonst kommt sie auf das angefangene "Blatt" .
Y	(REAL*8 Feld)	Die Abszissen bzw.
G	(REAL*8 Feld)	die Ordinaten der Funktion .
MODUS	(CHARACTER*8)	Schreib-Modus , s. Routine SERDIO .

Größen , die nur formal vorhanden sein müssen:

YMI,YMA	(REAL*8)	Bereichsgrenzen der Abszissen
GMI,GMA	(REAL*8)	bzw. der Ordinaten .
YED	(REAL*8)	Länge einer "Skala"
GED	(REAL*8)	auf der X- bzw. F-Achse .
NAMX	(CHARACTER*8)	Name für die X- bzw. für
NAMF	(CHARACTER*8)	die F-Achse (z.B. ' ZEIT ').

MASX	(CHARACTER*8)	Einheit für die X- bzw. für
MASF	(CHARACTER*8)	die F-Achse (z.B. ' SEC ').

Bemerkung zur NP :
die Zahl der Elemente aller Funktionen eines Blattes , NP wird durch die erste eingetragene Funktion entschieden. Nachfolgend geschriebene Funktionen werden - bei einer abweichenden NPi - entweder verkürzt , falls NPi > NP ist , oder - im Falle NPi < NP - die fehlende Werte mit dem letzten Funktionswert F(NPi) überschrieben.

```
                          ++++++++++++++++++++
Bei einer Datei mit der   + PLOT = 'PLOTEASY' +      Norm
                          ++++++++++++++++++++
```

wird die Subroutine SDINPY benutzt.

Aufruf : CALL SDINPY(KLASSE,NUMMER,NP,X,F,NAMF,MODUS) .

Hier werden folgenden Größen beim Aufruf benötigt:

KLASSE	(INTEGER)	2 Kenngrößen zur Identifikation
NUMMER	(INTEGER)	der Funktion .
NP	(INTEGER)	Anzahl der Funktions-Elemente , NP muß ≤ 1000 sein !
X	(REAL*4 Feld)	Die Abszissen bzw.
F	(REAL*4 Feld)	die Ordinaten der Funktion .
NAMF	(CHARACTER*8)	Name für die F-Achse (z.B. ' DRUCK ').
MODUS	(CHARACTER*8)	Schreib-Modus , s. Routine SERDIO .

5.3 L e s e n a u s e i n e r D a t e i .

Auch beim Lesen einer Datei muß die Leseroutine dem jeweiligen Datei-Norm entsprechen.

```
                    ++++++++++++++++++++
Bei einer           + GRA4 = 'GRAPHIC4' +      Datei
                    ++++++++++++++++++++
```

liest man mit der Subroutine SDEXG4 .

```
Aufruf : CALL SDEXG4(KLASSE,NUMMER,NP
                    ,X,XMI,XMA,XED,NAMX,MASX
                    ,F,FMI,FMA,FED,NAMF,MASF
                                       ,LOS)  .
```

Bei diesem Aufruf muß man allein LOS angeben :

LOS (INTEGER) Ordnungszahl der zu lesenden Funktion in der Datei.

Alle anderen Größen : KLASSE,NUMMER,NP ,X,XMI,XMA,XED,NAMX,MASX , F,FMI,FMA,FED,NAMF und MASF werden von der Leseroutine gefüllt . Ihre Beschreibung findet man in 5.2 , bei der Schreibroutine SDING4 .
Falls es in der Datei keine LOS-te Funktion gibt , dann erfolgt eine Fehlernachricht (KPRI im SERDIO-Aufruf muß > 0 sein !) und die Character-Größe NAMF wird zu '**ENDE**' gesetzt.

```
                    ++++++++++++++++++++
Bei einer           + GRA8 = 'GRAPHIC8' +      Datei
                    ++++++++++++++++++++
```

liest man mit der Subroutine SDEXG8 .

```
Aufruf : CALL SDEXG8(KLASSE,NUMMER,NP
                    ,Y,YMI,YMA,YED,NAMX,MASX
                    ,G,GMI,GMA,GED,NAMF,MASF
                                       ,LOS)  .
```

Auch bei diesem Aufruf muß man nur LOS angeben :

LOS (INTEGER) Ordnungszahl der zu lesenden Funktion in der Datei.

Alle anderen Größen : KLASSE,NUMMER,NP ,Y,YMI,YMA,YED,NAMX,MASX , G,GMI,GMA,GED,NAMF und MASF werden von der Leseroutine gefüllt . Für ihre Beschreibung s. die Schreibroutine SDING8 in 5.2 .
Falls die LOS-te Funktion nicht zu finden ist , erfolgt eine Fehlernachricht und NAMF wird zu '**ENDE**' gesetzt.

```
                    ++++++++++++++++++++
Bei einer           + FOL8 = '*FOLIA*8' +      Datei
                    ++++++++++++++++++++
```

liest man mit der Subroutine SDEXF8 .

```
Aufruf : CALL SDEXF8(KLASSE,NUMMER,NP
                    ,Y,YMI,YMA,YED,NAMX,MASX
                    ,G,GMI,GMA,GED,NAMF,MASF
                                       ,LOS,NEWP) .
```

Beim Aufruf muß man nur LOS angeben :

LOS (INTEGER) Ordnungszahl der zu lesenden Funktion in der Datei.

Alle anderen Größen : KLASSE,NUMMER,NP ,Y,YMI,YMA,YED,NAMX,MASX , G,GMI,GMA,GED,NAMF,MASF und NEWP werden von der Leseroutine gefüllt . Die Beschreibung dieser Größen steht in 5.2 bei der Routine SDINF8 .

bei NEWP gilt : NEWP enthält immer die Ordnungszahl des jeweiligen Blattes IBL . Bei einer Funktion, die am Anfang des Blattes steht , erscheint diese Zahl , bei anderen Funktionen wird -IBL zurückgegeben.

Bei einer erfolglosen Suche wird hier auch - nebst einer Fehlernachricht - NAMF zu '**ENDE**' gesetzt.

Bei einer ++++++++++++++++++++ Datei
+ PLOT = 'PLOTEASY' +
++++++++++++++++++++

liest man mit der Subroutine SDEXPY .

Aufruf : CALL SDEXPY(KLASSE,NUMMER,NP,X,F,NAMF,LOS) .

Anzugeben braucht man beim Aufruf nur LOS :

LOS (INTEGER) Ordnungszahl der zu lesenden Funktion in der Datei.

Die Größen KLASSE,NUMMER,NP,X,F und NAMF (Beschreibung bei der Routine SDINPY) werden von SDEXPY geliefert. Bei nichtvorhandenen Funktionen erscheint nach dem Aufruf NAMF als '**ENDE**' .

5.4 B e n ö t i g t e S o n d e r - S t e u e r k a r t e n .

Damit der Benutzer bei einer Stapel-Job Funktionen in eine SERVUS--Datei schreiben und/oder Funktionen aus einer Datei lesen kann , muß er beim Hintergrund-Job zusätzliche JCL-Karten ausfüllen .

Um die Dateien zu allokieren , braucht er folgende DD-Karten:

```
//G.FTxxF001 DD DSN=inr000.ergebnis.DATA,DISP=(OLD,SHR)
//G.FTyyF001 DD DSN=inr000.inform.DATA,DISP=(OLD,SHR)
```

Hier ist z.B. xx = IWR die symbolische Nummer der Schreib-Datei und yy=IRE die der Lese-Datei . Diese beiden Dateien müssen immer mit verschiedenen symbolischen Nummern allokiert sein.
Außerdem muß er den Modul INR105.SERVUS.LOAD(SERDIO) dem Job zufügen. Dies kann man z.B. mit den folgenden L-Karten erreichen :

```
//L. ...
//L.bibl DD DISP=SHR,DSN=INR105.SERVUS.LOAD
 INCLUDE bibl(SERDIO)
 ENTRY MAIN
//*
//G. ...
```

5.5 B e i s p i e l e .

Das erste Beispiel zeigt einen Stapel-Job , der eine Anzahl von Funktionen errechnet und sie anschließend in die Datei REFORM.DATA einträgt. Eine Lesedatei wird nicht benötigt.

```
//inr000B1 JOB (0abc,xyz,p9x9y),benutzer,MSGLEVEL=(0,0)
// EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C
      REAL*8 Y(1000),G(1000),YMI,YMA,YED,GMI,GMA,GED
      CHARACTER*8 NAMX,MASX,NAMF,MASF,NORMR,NORMW,MODUS
C
C     ************** E I N G A B E ****************
      DATA IRE/0/,IWR/16/,MODUS/'**OLD***'/,KPRI/2/
     =,KLASSE/1/,NP/50/
     =,NAMX/'   X    '/,MASX/' - 1 -  '/
     =,NAMF/'SIN(I*X)'/,MASF/' - 1 -  '/
C     ********************************************
C
      CALL SERDIO(IRE,NORMR,IWR,NORMW,MODUS,KPRI)
      IF(NORMW .NE. 'GRAPHIC8')                    GOTO 100
C
      Y(1)=0.D-0
      YMI = Y(1)
      DO 11 J=2,NP
   11 Y(J)=Y(J-1)+0.5D-1
      YMA=Y(NP)+0.5D-1
      GMA=1.D-0
      GMI=-GMA
      YED=0.2*(YMA-YMI)
      GED=0.2*(GMA-GMI)
      DO 51 NUMMER=1,5
      DO 21 J=2,NP
   21 G(J)=SIN(NUMMER*Y(J))
      IF(NUMMER .EQ. 1)       NAMF(5:5)='1'
      IF(NUMMER .EQ. 2)       NAMF(5:5)='2'
      IF(NUMMER .EQ. 3)       NAMF(5:5)='3'
      IF(NUMMER .EQ. 4)       NAMF(5:5)='4'
      IF(NUMMER .EQ. 5)       NAMF(5:5)='5'
      CALL SDING8(KLASSE,NUMMER,NP
     =,Y,YMI,YMA,YED,NAMX,MASX,G,GMI,GMA,GED,NAMF,MASF,MODUS)
   51 CONTINUE
C
  100 STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO)
 ENTRY MAIN
//*
```

```
//G.FT16F001 DD DSN=inr000.reform.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
//
```

Der Job meldet sich mit folgender Nachricht ab:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                                    +
+   25. 7.84    THE GRAPHS WILL BE COPIED INTO THE DATASET    REFORM  .DATA         +
+                                                                                    +
+            STORAGE-NORM IS      "GRAPHIC8"      MODUS = **OLD***                   +
+                                                                                    +
+     DATE OF THE ALLOCATION:   25. 1.84    LAST MODIFICATION   17. 2.84             +
+                                                                                    +
+                       THE DATASET CONTAINS    13 GRAPHS                            +
+                                                                                    +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

===>   14   GRAPH(   1/    1)  X : +   X    + - 1 -  +  F : +SIN(1*X)+ - 1 -  +
X1= 0.0       ,X( 50)=  2.450     ,  0.0        < X <  2.500     , U=0.50
F1= 0.0       ,F( 50)= 0.6378     ,  -1.000     < F <  1.000     , U=0.40

===>   15   GRAPH(   1/    2)  X : +   X    + - 1 -  +  F : +SIN(2*X)+ - 1 -  +
X1= 0.0       ,X( 50)=  2.450     ,  0.0        < X <  2.500     , U=0.50
F1= 0.0       ,F( 50)=-0.9825     ,  -1.000     < F <  1.000     , U=0.40

===>   16   GRAPH(   1/    3)  X : +   X    + - 1 -  +  F : +SIN(3*X)+ - 1 -  +
X1= 0.0       ,X( 50)=  2.450     ,  0.0        < X <  2.500     , U=0.50
F1= 0.0       ,F( 50)= 0.8757     ,  -1.000     < F <  1.000     , U=0.40

===>   17   GRAPH(   1/    4)  X : +   X    + - 1 -  +  F : +SIN(4*X)+ - 1 -  +
X1= 0.0       ,X( 50)=  2.450     ,  0.0        < X <  2.500     , U=0.50
F1= 0.0       ,F( 50)=-0.3665     ,  -1.000     < F <  1.000     , U=0.40

===>   18   GRAPH(   1/    5)  X : +   X    + - 1 -  +  F : +SIN(5*X)+ - 1 -  +
X1= 0.0       ,X( 50)=  2.450     ,  0.0        < X <  2.500     , U=0.50
F1= 0.0       ,F( 50)=-0.3111     ,  -1.000     < F <  1.000     , U=0.40
```

Das zweite Beispiel zeigt einen Stapel-Job , der eine Anzahl von Funktionen von einer Datei liest , diese mit sich selber multipliziert und die so gewonnenen , neuen Funktionen am Ende derselben Datei einspeichert.

```
//inr000B2 JOB (0abc,xyz,p9x9y),benutzer,MSGLEVEL=(0,0)
// EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C
      REAL*8 Y(1000),G(1000),F(1000),YMI,YMA,YED,GMI,GMA,GED
      CHARACTER*8 NAMX,MASX,NAMF,MASF,NORMR,NORMW,MODUS,NAMN
C
C     ************** E I N G A B E *****************
      DATA IRE/15/,IWR/16/,MODUS/'**OLD***'/,KPRI/2/
     =,KLASSN/2/,NAMN/'SI2(I*X)'/,LOS/14/
C     *********************************************
C
      CALL SERDIO(IRE,NORMR,IWR,NORMW,MODUS,KPRI)
      IF(NORMW .NE. 'GRAPHIC8')                    GOTO 100
C
      DO 51 NUMMNE=1,5
      CALL SDEXG8(KLASSE,NUMMER,NP
     =,Y,YMI,YMA,YED,NAMX,MASX,G,GMI,GMA,GED,NAMF,MASF,LOS)
      IF(NAMF .EQ. '**ENDE**')                     GOTO 100
      DO 21 J=2,NP
   21 F(J)=G(J)*G(J)
      IF(NUMMNE .EQ. 1)       NAMN(5:5)='1'
      IF(NUMMNE .EQ. 2)       NAMN(5:5)='2'
      IF(NUMMNE .EQ. 3)       NAMN(5:5)='3'
      IF(NUMMNE .EQ. 4)       NAMN(5:5)='4'
      IF(NUMMNE .EQ. 5)       NAMN(5:5)='5'
      CALL SDING8(KLASSN,NUMMNE,NP
     =,Y,YMI,YMA,YED,NAMX,MASX,F,GMI,GMA,GED,NAMN,MASF,MODUS)
   51 LOS=LOS+1
C
  100 STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.reform.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD DSN=inr000.reform.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
//
```

Der Job erzeugt folgende Nachrichten :

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+     25. 7.84      LIST OF THE GRAPHS OF THE DATASET        REFORM  .DATA     +
+                                                                              +
+                          STORAGE-NORM IS       "GRAPHIC8"                    +
+                                                                              +
+     DATE OF THE ALLOCATION:    25. 1.84     LAST MODIFICATION    17. 2.84    +
+                                                                              +
+                          THE DATASET CONTAINS     18 GRAPHS                  +
+                                                                              +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+   25. 7.84     THE GRAPHS WILL BE COPIED INTO THE DATASET     REFORM  .DATA   +
+                                                                              +
+             STORAGE-NORM IS       "GRAPHIC8"       MODUS = **OLD***          +
+                                                                              +
+     DATE OF THE ALLOCATION:    25. 1.84     LAST MODIFICATION    17. 2.84    +
+                                                                              +
+                          THE DATASET CONTAINS     18 GRAPHS                  +
+                                                                              +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
 #     14   GRAPH(    1/     1)  X : +    X     + - 1 -  +  F : +SIN(1*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)= 0.6378      ,  -1.000       < F <  1.000      , U=0.40

===>   19   GRAPH(    2/     1)  X : +    X     + - 1 -  +  F : +SI2(1*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)= 0.4067      ,  -1.000       < F <  1.000      , U=0.40

 #     15   GRAPH(    1/     2)  X : +    X     + - 1 -  +  F : +SIN(2*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)=-0.9825      ,  -1.000       < F <  1.000      , U=0.40

===>   20   GRAPH(    2/     2)  X : +    X     + - 1 -  +  F : +SI2(2*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)= 0.9652      ,  -1.000       < F <  1.000      , U=0.40

 #     16   GRAPH(    1/     3)  X : +    X     + - 1 -  +  F : +SIN(3*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)= 0.8757      ,  -1.000       < F <  1.000      , U=0.40

===>   21   GRAPH(    2/     3)  X : +    X     + - 1 -  +  F : +SI2(3*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)= 0.7668      ,  -1.000       < F <  1.000      , U=0.40

 #     17   GRAPH(    1/     4)  X : +    X     + - 1 -  +  F : +SIN(4*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)=-0.3665      ,  -1.000       < F <  1.000      , U=0.40

===>   22   GRAPH(    2/     4)  X : +    X     + - 1 -  +  F : +SI2(4*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)= 0.1343      ,  -1.000       < F <  1.000      , U=0.40

 #     18   GRAPH(    1/     5)  X : +    X     + - 1 -  +  F : +SIN(5*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)=-0.3111      ,  -1.000       < F <  1.000      , U=0.40

===>   23   GRAPH(    2/     5)  X : +    X     + - 1 -  +  F : +SI2(5*X)+ - 1 -  +
X1= 0.0        ,X( 50)=  2.450      ,  0.0          < X <  2.500      , U=0.50
F1= 0.0        ,F( 50)= 0.9680D-01  ,  -1.000       < F <  1.000      , U=0.40
```

6. Eine SERVUS-Datei wird gesichtet und kopiert .

6.1 Man kann von den Funktionen , die in einer SERVUS-Datei aufgezeichnet sind , einen Katalog bekommen. Dieser Katalog listet die Namenszeilen der in der Datei registrierten Funktionen der Reihe nach auf . Die Daten der Namenszeile erscheinen in folgender Ordnung:

Erste Druckzeile : Ordnungzahl der Funktion auf der Datei ,
KLASSE , NUMMER , NAMX , MASX , NAMF, MASF ,
zweite Druckzeile : X(1) , NP , X(NP) , XMI , XMA , XED ,
dritte Druckzeile : F(1) , NP , F(NP) , FMI , FMA , FED
(s. z.B. Abb. 1).

Man kann auch die SERVUS-Dateien - im Gegensatz zu normalen "DATA"--Dateien - kopieren , allerdings nur in andere SERVUS-Dateien .
Beim Kopieren besteht auch die Möglichkeit , die Daten im gewissen Umfang zu verändern . Diese Änderungen erfolgen dann aber für alle Funktionen , die kopiert wurden . Man kann

- die Anzahl der Funktionselemente ändern ,
- die Funktionen in ein vorgegebenes Datenfenster einfügen ,
- die Zahlenwerte der Funktionen einschränken.

Bei der Änderung der Punktezahl kann man die Funktion sowohl "verdichten", als auch "ausdünnen". Die neue Abszissen sind im jedem Falle äquidistant verteilt . Die neue Ordinaten werden in der Regel mit einem SPLINE-5-Verfahren aus den alten Werten ermittelt. Man kann aber auch lineare Interpolation benutzen.

Beim Zurecht-Schneiden der Funktionen durch ein vorgegebenes Datenfenster werden Elemente, deren Abszissen in das "Abszissen-Fenster" [XMIS ≤ X ≤ XMAS] nicht hineinpassen , beim Kopieren weggelassen . Außerdem werden Ordinaten-Werte , die aus einem "Ordinaten-Fenster" [FMIS ≤ F ≤ FMAS] hinausragen , durch die Werte FMIS oder FMAS ersetzt . Die Eingangswerte der Größen XMI , XMA , FMI , FMA ersetzt man dabei mit XMIS , XMAS , FMIS , FMAS .
Falls kein sinnvolles Fenster vorgelegt wurde , ermittelt das Programm - anstatt zurechtzuschneiden - ein minimales Datenfenster.

Die Einschränkung der Zahlenwerte erfolgt , indem man die Abszissen

und/oder Ordinaten so lange mit dem Faktor 0.001 oder 1000. multipliziert , bis die Beträge aller diesen Zahlen im Bereich (0.001,1000.) liegen . Die Faktoren bleiben dabei im Bereich [1.E-75,1.E+75].
Diese Umnormierung wird dann auch in dem Namen MASX und/oder MASF vermerkt : das Eingangs-Wort der jeweiligen Größe wird durch das Wort ' *1.E+yz' (= 1/Umnormierungsfaktor) ersetzt .

6.2 S i c h t e n b z w. K o p i e r e n im S t a p e l - B e t r i e b .

Zum Sichten/Kopieren im Stapel-Betrieb kann man die Hintergrund--Prozedur KOPIERT benutzen . Sie sieht wie folgt aus:

Name : tso000.SERVUS.CNTL(KOPIERT)

```
//inr000k  JOB (0abc,xyz,p9x9y),benutzer,MSGLEVEL=(1,1)
// EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C               PROGRAMM KOPIERT
      CALL COPYDA
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT14F001 DD DSN=inr000.bilder.DATA,DISP=(OLD,KEEP)
//G.FT12F001 DD DSN=inr000.vielbild.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LAKOPI IRE=14,IWR=12,MODUS='**OLD***',INLIST=0,LAUF=1,31,
 LAX=0,LAF=0,KPRI=1,NUPOIN=30, &END
 &LAKOPI IRE=12,IWR=0,INLIST=0,LAUF=1,99,KPRI=2, &END
 &LAKOPI IRE=0, &END
//
```

Eingabebeschreibung:
Die Daten werden im NAMELIST-Format eingegeben . Die Liste der Eingabedaten lautet :

```
/LAKOPI/ IRE,IWR,MODUS,INLIST,LAUF,KPRI,LAX,LAF,XMIS,XMAS,XEDS,
         FMIS,FMAS,FEDS,IFILL,NUPOIN
```

Von den Listen-Größen sind XMIS,XMAS,XEDS,FMIS,FMAS,FEDS REAL*4 Zahlen , MODUS ist ein CHARACTER*8 Name , alle anderen Zahlen sind INTEGERs . LAUF ist ein INTEGER-Feld mit 50 Elementen .
Die Bedeutung der einzelnen Eingabe-Größen:

IRE , IWR — sind die symbolischen Nummern der Lese- bzw Schreib--Datei . Falls IWR = 0 oder IWR = IRE ist , wird nicht geschrieben , es wird nur die Datei IRE gelesen . Eine Eingabe-Liste mit IRE = 0 stoppt die Prozedur .

INLIST , LAUF — bestimmen zusammen eine Menge von Ordnungszahlen . Gelesen werden dann die Funktionen , die in der Lese-Datei zu diesen Ordnungszahlen gehören . Falls

INLIST > 0 ist , besteht die Zahlenmenge aus den Ordnungszahlen { LAUF(1),LAUF(2),...,LAUF(INLIST) } , falls

INLIST = 0 ist , sind es die Ordnungszahlen { LAUF(1) ≤ i ≤ LAUF(2) } bzw. { LAUF(1) ≥ i ≥ LAUF(2) } und falls

INLIST < 0 ist , hat die Zahlenmenge die -INLIST Elemente { LAUF(1),[LAUF(1)+LAUF(2)],[LAUF(1)+2*LAUF(2)],... }.

KPRI — ist ein Maß für die Intensität der Ausgabe :

Für KPRI = 0 wird nichts ausgedruckt ,

für KPRI = 1 werden die Namenszeilen der notierten Funktionen ausgedruckt ,

für KPRI = 2 werden auch die Namenszeilen der gelesenen Funktion ausgedruckt .

KPRI > 2 wird nicht empfohlen . Es werden zu viele Daten ausgedruckt und es kann zur Zeilen-Überschreitung kommen .

Die folgenden Eigabedaten werden nur dann gebraucht , falls in eine Datei geschrieben wird (IWR > 0) .

MODUS — Schreib-Modus . Er kann folgende Werte haben:

='**OLD***' das Schreiben in die Datei beginnt hinter dem letzten Eintrag , d.h. die alten Aufzeichnungen bleiben erhalten !

='**NEW***' das Schreiben in die Datei beginnt am Anfang , hinter der Kopfzeile , d.h. die alten Aufzeichnungen werden zerstört!

='*NEWPAG*' das Schreiben in die Datei beginnt hinter dem letzten Eintrag . Bei einer FOL8-Datei wird außerdem - im Gegensatz zum OLD-Modus - ein neues Blatt aufgeschlagen .

LAX Korrektur der Abszissen beim Kopieren ?

Bei LAX = 0 werden die Abszissen unverändert kopiert. Falls
sie = 1 ist , dann werden die Funktion durch das vorgegebene
oder =-1 Abszissen-Fenster eingeschränkt und falls
LAX > 0 ist , dann werden auch die Zahlenwerte der Abszissen in den (0.001,1000.)-Bereich überführt.

LAF Korrektur der Ordinaten beim Kopieren ?

Bei LAF = 0 werden die Ordinaten unverändert kopiert. Falls
sie = 1 ist , dann werden die Ordinaten mit dem vorgegebenen
oder =-1 Ordinaten-Fenster zurechtgeschnitten und falls
LAF > 0 ist , dann werden auch die Zahlenwerte der Ordinaten auf den (0.001,1000.)-Bereich eingeschränkt.

XMIS , XMAS Abszissen-Fenster . Falls man XMIS ≥ XMAS eingibt , bestimmt die Prozedur das optimale Fenster anhand der Abszissenmenge selbständig .

FMIS , FMAS Ordinaten-Fenster . FMIS ≥ FMAS erzeugt ein optimales Ordinaten-Fenster .

XEDS , FEDS vorgegebenen Skalen-Längen auf der X- bzw. F- Achse (s. S. 11).

IFILL falls IFILL > 0 ist , wird die vorgegebene X-Fenster ganz mit der Funktion aufgefüllt , indem man den ersten bzw. den letzten Funktionswert - soweit noch Platz vorhanden - fortsetzt.

NUPOIN Veränderte Punktzahl. Falls

-2< NUPOIN <2 ist , wird die alte Punktzahl beibehalten.

Sonst wird aus dem alten eine neue Funktion mit ABS(NUPOIN) Elementen erzeugt . Die neue Ordinaten werden bei

NUPOIN < 0 mit linearer Interpolation und bei

NUPOIN > 0 mit dem SPLINE-5-Verfahren berechnet.

Eingabe-Beispiele :

```
 &LAKOPI IRE=12,IWR=0,INLIST=0,LAUF=1,99,KPRI=2, &END
```

Teilkatalog der SERVUS-Datei "12" , von der 1. bis zum 99. Funktion. Als Ausgabe-Beispiel s. Abb. 1 .

```
 &LAKOPI IRE=13,IWR=16,MODUS='**NEW***',KPRI=0,LAUF=13,20,INLIST=-90,
 LAX=1,LAF=1,XMIST=1.0,XMAST=50.,FMIST=-2.E+2,FMAST=0.0E+2, &END
```

90 Funktionen der Datei "13" werden in die Datei "16" kopiert und zwar die 13. , die 33. , usw. Die Datei "16" wird neu beschrieben ! Es erfolgt keine Druckausgabe . Sowohl die Abszissen , als auch die Ordinaten werden eingepasst und eingeschränkt.

```
 &LAKOPI    IRE=14,IWR=12,MODUS='**OLD***',INLIST=0,LAUF=1,31,KPRI=0,
 LAX=0,LAF=0,NUPOIN=30, &END
```

Die ersten 31 Funktionen der Datei "14" werden in die Datei "12" kopiert.Beim Kopieren wird auch die Punktezahl geändert: die kopierte Funktionen bestehen alle aus 30 äquidistanten Elementen , die durch ein SPLINE-5-Verfahren erzeugt wurden.

```
 &LAKOPI IRE=0, &END
```

Immer erforderliche End-Eingabe. Sie ist stets die letzte NAMELIST.

6.3 S i c h t e n / K o p i e r e n i m D i a l o g .

Es gibt auch ein Dialog-Verfahren zur Durchsicht und zum eventuellen Kopieren einer Datei am Bildschirm : DATAS
DATAS stellt bis zu drei Dateien zum Lesen bereit und eine vierte zum Schreiben . Die Angaben über diese Dateien werden in einer Kommando-Dialog erfragt . Anschließend setzt ein FORTRAN-Gespräch ein , in dem die Eingabedaten zum Kopieren - Listen erfragt werden. Diese Eingabedaten entsprechen den Eingabedaten der Stapelprozedur.
Der Aufruf lautet :

```
EX 'tso000.SERVUS.CLIST(DATAS)' .
```

Die Prozedurliste ist im Kap. 11 abgedruckt.

7. Einige Funktionen einer SERVUS-Datei werden gestrichen .

Man kann Funktionen , die in einer SERVUS-Datei nicht mehr erwünscht sind , mit Hilfe der Hintergrund-Prozedur LOESCHT eliminieren . Die Prozedur kopiert die betroffene Datei in eine Hilfsdatei , wobei die unerwünschten Funktionen unberücksichtigt bleiben . Die Hilfsdatei wird dann wieder in die ursprüngliche Datei zurückkopiert . Von der bereinigten Datei wird anschließend ein neuer Katalog erstellt .
Anstatt die fragliche Funktionen zu löschen , kann man sie im Rahmen dieser Prozedur auch renormieren , d.h. die Ordinaten aller betroffenen Funktionen mit einem Faktor multiplizieren.

Die Hintergrund-Prozedur (mit anschließendem Katalogschritt) sieht aus wie folgt:

Name : tso000.SERVUS.CNTL(LOESCHT)

```
//inr000e  JOB (0abc,xyz,p9x9y),benutzer,MSGLEVEL=(1,1)
//ERASE  EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C                PROGRAMM LOESCHT
      CALL ERASE
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.vielbild.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
//G.SYSIN DD *
 &LALOES IRE=15,IWR=16,NUMMAX=300,NIX=219,220,KPRI=0, &END
 &LALOES IRE=0, &END
//*
//KATALOG EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C                    KATALOG
      CALL COPYDA
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.vielbild.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
```

```
 &LAKOPI IRE=15,IWR=0,MODUS='**OLD***',INLIST=0,LAUF=1,1000,
 KPRI=2, &END
 &LAKOPI IRE=0, &END
//
```

Eingabebeschreibung:
Die Daten werden im NAMELIST-Format eingegeben . Die Liste ist die folgende:

```
NAMELIST /LALOES/ NUMMAX,NIX,IRE,IWR,KPRI,FAKTOR
```

Alle Listen-Größen außer FAKTOR sind INTEGERs, NIX ist ein Feld von 50 Elementen , FAKTOR ist eine REAL*4-Zahl.
Die Bedeutung der einzelnen Eingabe-Größen:

NUMMAX	ist die Ordnungszahl der letzten Funktion , die bei der Prozedur noch erhalten bleiben soll .
NIX(1),NIX(2), ...,NIX(k),...	sind die Indizes der Funktionen , die gelöscht oder werden sollen .
IRE	ist die symbolische Nummer der zu korrigierenden Datei. Eine Liste mit IRE = 0 beendet die Prozedur.
IWR	ist die symbolische Nummer der Hilfsdatei.
KPRI	Druckintensität , es empfiehlt sich KPRI = 0.
FAKTOR	ist ein Renormierungsfaktor. Falls
= 0.0	ist , dann werden die angegebene Funktionen gelöscht,
sonst	werden die Ordinaten der angegebenen Funktionen mit FAKTOR multipliziert.

8. Funktionen aus einer SERVUS-Datei werden ausgedruckt .

Funktionen , die in einer SERVUS-Datei aufgezeichnet sind , kann man mit dem Schnelldrucker - im Rahmen eines Hintergrund-Jobs - ausdrucken lassen .
Die Funktionen werden als Zahlen-Kolonnen ausgedruckt : die Ordinaten einer Funktion erscheinen untereinander . Die Kolonne kann entweder mit dem ersten , oder mit dem letzten Funktions-Element beginnen . Auf jede Druckseite , über der Kolonne werden - als Beschriftung - beide Namen NAMF und MASF der betroffenen Funktion sowie die Kenngrößen KLASSE , NUMMER ausgedruckt (s. Abb. 3) .
Man kann sich darauf beschränken , nur die Ordinaten einer Funktion auszudrucken . In diesem Fall können bis zu 8 Funktionen nebeneinander auf einer Druckseite Platz finden . Man kann aber auch eine Abszissen-Kolonne am linken Rand des Druckseite ausgeben , oder für jede Funktion Abszissen und Ordinaten ausdrucken . In diesem letzten Falle passen dann nur noch 4 Funktionen auf eine Seite .
Die Genauigkeit der Ausgabe richtet sich nach der auszudruckenden Datei ; GRA8 und FOL8 Dateien werden mit 8-Stellen ausgedruckt , die anderen mit 6 .
Die Hintergrund-Prozedur DRUCKT sieht aus wie folgt:

Name : tso000.SERVUS.CNTL(DRUCKT)

```
//inr000p  JOB (0abc,xyz,p9x9y),benutzer,MSGLEVEL=(1,1)
// EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C               PROGRAMM DRUCKT
      CALL PAGEFF
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,NEXT)
 ENTRY MAIN
//*
//G.FT16F001 DD DSN=INR000.VIELBILD.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LADRUC IRE=16,INLIST=0,LAUF=220,225,MODPRI='UP.EINX1',KPRI=0, &END
 &LADRUC IRE=0, &END
//
```

Eingabebeschreibung:
Die Daten werden im NAMELIST-Format eingegeben . Die Liste lautet :

/LADRUC/ IRE,INLIST,LAUF,MODPRI,JOAN,JOEN,JO,KPRI

Bei den Listen-Größen ist nur MODPRI ein CHARACTER*8 Name , alle anderen Daten sind INTEGERs . LAUF ist ein INTEGER-Feld mit 50 Elementen .
Die Bedeutung der einzelnen Eingabe-Größen:

IRE — ist die symbolische Nummer der auszudruckenden Datei . Eine Eingabe-Liste mit IRE = 0 stoppt die Prozedur .

INLIST , LAUF — bestimmen zusammen eine Menge von Ordnungszahlen . Ausgedruckt werden dann die Funktionen , die in der Datei zu diesen Ordnungszahlen gehören . Falls

INLIST > 0 ist , besteht die Zahlenmenge aus den Zahlen { LAUF(1),LAUF(2),...,LAUF(INLIST) } . Bei dieser Eingabe kann man die Besetzung eines Blattes steuern : ein positives LAUF(j) bedeutet , daß mit der LAUF(j)-ten Funktion der Datei ein neues Blatt beginnt, ein negatives LAUF(j+1) bewirkt , daß die [-LAUF(j+1)]-te Funktion auf dem aufgeschlagenen Blatt ausgegeben wird , usw. LAUF(k) = 0 schließt die Druckseite ab .

INLIST = 0 entspricht der folgende Zahlenmenge : { LAUF(1) ≤ i ≤ LAUF(2) } bzw. { LAUF(1) ≥ i ≥ LAUF(2) } . Falls

INLIST < 0 ist , hat man -INLIST Ordnungszahlen : { LAUF(1),[LAUF(1)+LAUF(2)],[LAUF(1)+2*LAUF(2)],... }. Bei den beiden letzten Eingaben-Typen , INLIST = 0 und INLIST < 0 werden die Blätter "dicht gepackt" , d.h. es werden soviel Kolonnen wie möglich auf einen Blatt untergebracht .

MODPRI — steuert die Besonderheiten des Ausdruckes . Für

= 'DN......' wird mit dem ersten Element begonnen, für

= 'UP......' mit dem letzten Element der Funktion .

= '.......0' bedeutet , daß nur die Ordinaten gedruckt werden

= '.......1' daß die Abszissen der - auf dieser Seite - ersten Funktion auch erscheinen und bei

= '...sonst' werden zu jeder Funktion Abszissen und Ordinaten ausgedruckt .

JOAN,JOEN,JO "Rhythmus" des Ausdruckes . Der Druck beginnt mit dem JOAN-ten Element und hört mit dem JOEN-ten auf . In diesem Bereich wird jede JO-te Element ausgedruckt .

KPRI ist wieder ein Maß für die Intensität der Ausgabe . Die Funktions-Elemente werden hier aber auch bei KPRI=0 ausgegeben .

Der Ausdruck , der in den Abb. 2 - Abb. 4 dargestellt ist , entspricht der Eingabe :

```
&LADRUC IRE=16,MODPRI='DN.EINX1',INLIST=7,JOAN=1,JO=1,
LAUF=4,-5,-3,-6,-13,-14,-1, &END
```

9. Funktionen aus einer SERVUS-Datei werden als eine Kurvenschar gezeichnet.

9.1 Über das Abbildungsverfahren.

Die im folgenden (Kap. 9 und 10) beschriebene bildhafte Ausgabe der gespeicherten Funktionen beruht auf dem Graphischen System (GS) der HDI /2/ . Dabei werden zum Zeichnen , Schreiben und bei der Herstellung der Bild-Rahmen die im TRACEGS-System /4/ zusammengefaßten Routinen verwendet .
Die Anwendung des GS schließt die Möglichkeit , die gezeichneten Bilder zu speichern , mit ein . Diese Bildspeicherung erlaubt es dem Benutzer , von dem einmal gezeichneten Bild beliebig viele Abzüge in verschiedenen Größen und an verschiedenen Zeichengeräten zu fertigen. Außerdem kann man das gespeicherte Bild nachträglich - mit Hilfe von GS-Befehlen - ändern (z.B. die Unterschrift).
Für die Bildspeicherung braucht der Benutzer ein "Buch" - eine gegliederte Datei mit der Namens-Form 'inr000.gsbook' - in die nachher die Bilder als Glieder (Members) eingfügt werden (s. /2/ oder /4/) . Diese Datei kann man mit der folgenden TSO-Kommandofolge anlegen :

```
ATTRIB buch BLKSIZE(19069) RECFM(U)
ALLOC DA('inr000.gsbook') NEW UNIT(DISK) VOLUME(bat00c)  +
      SPACE(10,10) TRA DIR(5) USING(buch)
FREE ATTRLIST(buch)
```

Die kleingedruckten Namen , sowie den Raumbedarf (10,10) und (5) muß der Benutzer ausfüllen .
Die abzubildende Datei muß in GRA4 normiert sein !

9.2 Stapel-Verfahren für die Kurven-Darstellung.

Die Prozedur FIGURB erzeugt Kurvendarstellungen der gespeicherten Funktionen im Stapelbetrieb. In einem Auftrag kann mann bis zu 100 Bilder - Kurven-Darstellungen - erzeugen , die dann zu einem Gesamtbild zusammengelegt und durch eine der nachgeschalteten Zeichengeräte

(Xynetics - Versatec - Statos) ausgegeben werden . Man kann auch dieses Gesamtbild unter den Namen TOO1 in einem "Buch" speichern.
Der Benutzer kann das Aussehen der Bilder weitgehend nach eigenem Vorstellungen gestalten - oder aber auch Standard-Darstellungen verwenden , die sehr wenig Aufwand bei der Eingabe erfordern.

In der Regel - bei Mindesteingabe - werden zu jeder gezeichneten Funktion die Funktions-Namen NAMF(i) sowie die Kenngrößen KLASSE(i) und NUMMER(i) in das Bild mit-eingezeichnet . NAMF (und MASF) der ersten Funktion des Bildes werden - im Regelfall - neben die Ordinaten-Achse gedruckt . KLASSE(1) und NUMMER(1) erscheinen an einer - von dem Benutzer bestimmten - Stelle des Bildes. Die Kenn-Nummern und Namen der folgenden Funktionen (wenn es welche gibt) werden übereinander , in einer "wachsender" Kolonne ausgegeben und zwar über die Kenn-Nummer der ersten Funktion beginnend . Über die Abszissen--Achse werden NAMX und MASX der ersten Funktion geschrieben (s. Abb. 5) . In der Regel werden die einzelne Bilder im TOO1 mit 1 beginnend durchnumeriert und sie erhalten keine Bild-Unterschrift außer der Bildnummer. Die "Abb. 1" in Abb. 5 zeigt ein Bild , das mit Mindesteingabe erzeugt wurde (s. erste Eingabekarte bei der nachfolgend abgedruckten Prozedur).
Es besteht aber auch die Möglichkeit , Bildnummer und Bildunterschrift vorzugeben . Man kann auch für die einzelnen Kurven anstatt der Kenn-Nummern und Namen Beschriftungen vorgeben. Auch die Achsen--Beschriftungen kann man nach eigenen Vorstellungen gestalten. Außerdem kann man - mit Hilfe der Dateien tsoOOO.MEMORY(ABB1) ... ABB4) - Bildunterschriften mit Groß- und Kleinschrift und mit Sonderzeichen zeichnen zu lassen (Teilbild rechts oben in Abb. 5). Die Regeln zum Reinschreiben der Vorlagen in die obigen Dateien findet man im TRACEGS-Handbuch /4/ unter "TEXT" (S. 35).
Schließlich kann man - zusätzlich zu den Funktionen - eine Null-Linie in das Bild eintragen lassen .

Die Liste der Hintergrund-Prozedur FIGURB (mit dem VERSATEC-Gerät):

Name : tsoOOO.SERVUS.CNTL(FIGURB)

```
//inrOOOf  JOB (Oabc,xyz,p9x9y),benutzer,MSGLEVEL=(1,1),REGION=2536K
//A EXEC F7CLG,PLOT=GS7,USER='SYS2.TRACEGS7'
//C.SYSIN DD *
```

```
C     F I G U R B   ( STAPELVERSION VON FIGUR )
      CHARACTER*4 INST
      DATA IRE/15/,IDIN/6/,NETZ/1/,MEM/30/,KPRI/2/,INST/'INR '/
      CALL FIGURB(IRE,IDIN,NETZ,MEM,KPRI,INST)
      STOP
      END
//*
//L.BIBS DD DSN=INR105.SERVUS.LOAD,DISP=SHR
//L.SYSIN DD *
 INCLUDE BIBS(SERDIO,FIGURB,NEXT,PAPIER,FENST4,NARS)
 ENTRY MAIN
//*
//G.PLOTPARM DD *
 &PLOT XMAX=290.1,YMAX=23., &END
//G.COMM     DD SYSOUT=*
//G.FT15F001 DD DSN=inr000.vielbild.DATA,DISP=(OLD,KEEP)
//G.FT31F001 DD DSN=tso000.MEMORY(ABB1),DISP=(OLD,KEEP)
//G.FT32F001 DD DSN=tso000.MEMORY(ABB2),DISP=(OLD,KEEP)
//G.FT33F001 DD DSN=tso000.MEMORY(ABB3),DISP=(OLD,KEEP)
//G.FT34F001 DD DSN=tso000.MEMORY(ABB4),DISP=(OLD,KEEP)
//G.TRACEGS7 DD DSN=inr000.gsbook,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LAFIGU INLIST=1,LAUF=1, &END
 &LAFIGU INLIST=2,LAUF=7,-8,48*0,ABTEX='  ZWEITENS',NUL=1,NABB=5,
 FMIST=-5.,FMAST=5.,FEDST=1.,XZUS$=5.,FZUS$=5., &END
  ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
 &LAFIGU INLIST=-5,LAUF=18,18,48*0,ABTEX='  SECHSTENS',NUL=1,
 XZUS$=13.,FZUS$=1.0,FMIST=-150.,FMAST=150.,FEDST=50., &END
 &LAFIGU LAUF=90,18,48*0,ABTEX='  SIEBTENS',ITX=31,XRAR$=2.,FRAR$=4.,
 FAUF$=9.,FABB$=4.,TEXTX(47:67)='KANALHOEHE  ( IN CM )',
 TEXTF(10:40)='GASGESCHWINDIGKEITEN ( CM/SEC )',
 TEXTZ(1)(1:9)='( 3, 93 )',TEXTZ(2)(1:9)='( 3, 95 )',
 TEXTZ(3)(1:9)='( 3, 97 )',TEXTZ(4)(1:9)='( 3, 99 )',
 TEXTZ(5)(1:9)='( 3,101 )',IQ=0,30,10,5,-1, &END
  ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
 &LAFIGU LAUF(1)=0, &END
//*
//B EXEC SVPLOT,SPACE=50
//
```

Die abgedruckte Stapel-Prozedur holt die zu zeichnende Funktionen aus der Datei inr000.vielbild.DATA und speichert das fertige Bild (Abb. 5) in das "Buch" inr000.gsbook. Die Datei tso000.MEMORY(ABB1) wird bei der 4. Eingabe (ITX=31) gebraucht um die groß/klein-geschriebene Bildunterschrift zu übertagen.

Eingabebeschreibung:

Die Eingabe besteht aus allgemeinen Angaben zum Zeichnungs-Verfahren und aus speziellen Anweisungen für jedes einzelne Bild.
Die allgemeinen Verfahrens-Hinweise sind im Aufruf-Kopf der Prozedur

enthalten und man kann sie in der DATA-Anweisung ändern . Diese Steuergrößen sind:

(IRE,IDIN,NETZ,MEM,KPRI,INST).

Von diesen Größen ist nur INST ein CHARACTER*4 Wort , alle anderen sind INTEGERs . Sie haben die folgende Bedeutung:

IRE	ist die symbolische Nummer der SERVUS-Datei,
IDIN	ist die DIN-Größe des Bildes :
> 0	zeichnet das Bild in DIN-A-"IDIN"-Querformat und
< 0	in DIN-A-"-IDIN"-Hochformat.
NETZ > 0	erzeugt ein Trennungsnetz zwischen den Bildern. Falls
2 | -2	ist und ABS(IDIN) < 5 , dann erhält jedes Teilbild das KFK-Emblem (s. z.B. Abb. 8).
MEM	wählt den Zeichengerät: für
1 | 10	wird der STATOS-Plotter benutzt , für
2 | 20	der XYNETICS-Plotter und für
3 | 30	der VERSATEC-Plotter . Falls
MEM =	
10 | 20 | 30	ist , dann wird das Gesamtbild T001 auch gespeichert.
KPRI	ist die Ausdruck-Intensität des Begleit-Protokolls.
INST	4 Zeichen , die auf dem KfK-Emblem mit erscheinen (Institutskürzel , +INR + auf der Abb. 8).

Die speziellen Bild-Informationen werden im NAMELIST-Format eingegeben , für jedes Teil-Bild eine Liste "LAFIGU" :

```
/LAFIGU/ INLIST,LAUF,XMIST,XMAST,XEDST,FMIST,FMAST,FEDST,
         SCR$,XABB$,FABB$,XAUF$,FAUF$,XRAR$,FRAR$,XZUS$,FZUS$,
         TAX$,TAF$,TAZ$,NABB,NUL,ITX,IQ,IQS,ABTEX,TEXTX,TEXTF,TEXTZ
```

Von den Listen-Größen sind ABTEX , TEXTX und TEXTF CHARACTER*70 Namen , TEXTZ ist ein Feld von 20 CHARACTER*70 Elementen , NABB , INLIST, ITX , IQS und NUL sind INTEGERs , IQ ist ein 20-er Feld und LAUF ist ein 50-er Feld von INTEGERs . Alle anderen Daten sind REAL*4 Zahlen .
Die Bedeutung der einzelnen Listen-Größen:

INLIST , LAUF dürfen auch bei Mindesteingabe nicht fehlen. Sie bestimmen zusammen die Zahlenmenge der zu zeichnenden Funktionen. Falls

INLIST > 0 ist , besteht sie aus den Zahlen
{ LAUF(1),LAUF(2),...,LAUF(INLIST) } .
Bei dieser Eingabe kommt nach der ersten Funktion eine weitere nur dann in das angefangene Bild , falls das zugehörige LAUF(j) negativ ist . Ein Index LAUF(k) ≥ 0 beendet das Bild . Falls

INLIST = 0 ist , ist die Zahlenmenge die folgende :
{ LAUF(1) ≤ i ≤ LAUF(2) } bzw.
{ LAUF(1) ≥ i ≥ LAUF(2) } und falls

INLIST < 0 ist , hat man die -INLIST Ordnungszahlen :
{ LAUF(1),[LAUF(1)+LAUF(2)],[LAUF(1)+2*LAUF(2)],... }.
Bei den Eingabe-Typen INLIST = 0 und INLIST < 0 werden - im Gegensatz zu dem "INLIST > 0"-Fall - alle Funktionen , die durch die Zahlenmenge bestimmt sind , in dasselbe Bild eingezeichnet .

LAUF(1) = 0 : eine Eingabe-Liste mit LAUF(1) = 0 stoppt die Prozedur FIGURB .

Die folgenden Daten umreissen das - für das Bild vorgesehene - Datenfenster (s. auch Kap. 2.3 und 6.2):

XMIST,XMAST bestimmen den erlaubten Werte-Bereich auf der X-Achse,
FMIST,FMAST den entsprechenden Bereich auf der F-Achse.
XEDST,FEDST sind die vorgegebenen Skalenlängen (s. S. 11).

Falls bei einer Eingabe XMIST ≥ XMAST ist , oder XEDST = 0 , dann benutzt die Prozedur anstelle dieser Größen XMI , XMA und XED von der ersten Funktion des Bildes. Falls auch diese Größen ungeeignet sind , dann werden diese Daten anhand der Abszissenmenge von der Prozedur ermittelt . Entsprechendes gilt bei der Größen FMIST , FMAST und FEDST .

Die folgenden Daten bestimmen das Aussehen des Bildes:

SCR$ ist die Höhe der verwendeten Schriftzeichen.

XAUF$, FAUF$ sind die Koordinaten der linken , unteren Ecke des inneren Rahmens,

XRAR$, FRAR$ die Abstände zum Bildrand am rechten , bzw. am oberen Ende des inneren Rahmens.

XZUS$, FZUS$ bestimmen den Anfang der Kolonne , in die die Kennungen der einzelnen Funktionen kommen , d.h. die Stelle , wo KLASSE(1) , NUMMER(1) hingeschrieben wird.

XABB$, FABB$ sind die Anfangs-Koordinaten der Bildunterschrift.

Alle hier beschriebenen Größen SCR$,XAUF$ - ... - FABB$ sind in den relativen Einheiten UNITX bzw. UNITF anzugeben : UNITX ist 1/32. der X-Ausdehnung des Bildes , UNITF ist 1/32. der F-Ausdehnung .

TAX$ ist der Abstand der X-Achsen-Beschriftung über der X-Achse in Einheiten der Schrifthöhe. Bei TAX$ < 0 kommt die Schrift unter die X-Achse.

TAF$ ist der Abstand der F-Achsen-Beschriftung rechts von dieser Achse (in Schrifthöhe) . Bei TAF$ < 0 liegt die Schrift links von der F-Achse.

TAZ$ in Schrifthöhe-Maß legt die Größe der Kurven--Markierungen fest.

Die Abb. 6 zeigt die Bedeutung der einzelnen Größen XAUF$ ... .

Die restlichen Daten beziehen sich auf die Beschriftungen und auf die Kurvendarstellung.

NABB ist die vorgesehene Bild-Nummer . Falls

= 0 ist , werden die Bilder durchnumeriert .

NUL > 0 erzeugt eine Null-Linie zusätzlich .

ITX entscheidet über die Beschriftung :

ITX = 0 erzeugt ein Bild , bei der nur ABTEX und die Funktionsgrößen KLASSE(i),NUMMER(i),NAMF(i),MASF(1) und MASX(1) zum Beschriften benutzt werden. Bei

ITX > 0 werden die Achsen mit TEXTX bzw TEXTF beschriftet und die Kurven mit den entsprechenden TEXTZ(i). Falls hierbei

ITX = 30+j ist, mit 0 < j < 5 , dann kommt die Bildunterschrift aus der der Datei MEMORY(ABBj) , ansonsten wird wieder

	ABTEX benutzt.
ABTEX	enthält die Bildunterschrift (≤ 70 Zeichen),
TEXTX und	enthalten die Beschriftungen für die X-
TEXTF	bzw. für die F-Achse (je ≤ 70 Zeichen) .
TEXTZ(1) , ..	ermöglichen es , die einzelnen Kurven
.. ,TEXTZ(20)	zu beschriften (je ≤ 70 Zeichen) .
IQ und IQS	wählen die Kurvendarstellung . Bei ITX > 0 wird IQ benutzt:
IQ(i) > 0	zeichnet die i-te Kurve als eine Linie , mit jeden IQ -tem Punkt markiert,
IQ(i) = 0	zieht eine unmarkierte Linie und
IQ(i) < 0	zeigt nur jeden IQ-ten Punkt der Funktion.
	Bei einer Liste mit ITX = 0 zählt IQS . Falls
IQS > 0	ist , wird auf der ersten Kurve jede IQS-te Punkt auf
	den zweiten jede IQS-1-te usw. markiert.
IQS = 0	erzeugt unmarkierte Kurven und bei
IQS < 0	werden alle Punkte ohne Verbindungen gezeigt.

Bemerkung zur Bildspeicherung : die durch die NAMELIST-Reihe hintereinander abgerufene Einzelbilder erscheinen am nachgeschalteten Zeichengerät als ein einziges Bild . Falls es zur Bildspeicherung kommt, wird dieses Gesamtbild gespeichert. Dieses gespeicherte Bild (Bildname = T001) ist wie folgt in Teilbilder aufgegliedert: das zuerst erzeugte Bild wird als

T001.RAHMEN01 , T001.INHALT01 , T001.TEXT01

aufbewahrt , das zweite als

T001.RAHMEN02 , T001.INHALT02 , T001.TEXT02 usw.

Die Bildteile INHALTOj erfahren eine weitere Unterteilung :

.INHALTOj = { .XTEXT , .FTEXT , .KURVEN , .KURVTX } .

In den Teil RAHMENOj kommt nur der äußere Rahmen des Teilbildes. Die Funktionen , der innere Rahmen , die Achsen- und die Kurvenbeschriftungen werden im Teil INHALTOj gespeichert. Die X-, bzw. F-

-Beschriftungen kommen dabei in .XTEXT , bzw. .FTEXT , die Funktionszüge in .KURVEN und die Kurvennamen in .KURVTX . Der Teil TEXTOj nimmt die Bildunterschrift auf.
Diese Gliederung erleichtert es, die gespeicherte Bilder mit Hilfe der Kommandosprache des GS /2/ nachträglich zu Korrigieren.

9.3 K u r v e n - D a r s t e l l u n g i m D i a l o g.

Funktionen einer SERVUS-Datei kann man auch im Dialogverfahren zu Bildern zusammenstellen und am Graphischen Anschluß (GA) anschauen. Der Vorteil dieses Verfahrens gegenüber dem Stapelbetrieb ist , daß man die Darstellungen, Beschriftungen, usw. sofort korrigieren kann , der Hauptnachteil ist , daß man in diesem Modus keinen direkten Zugang zu den Zeichengeräten hat.
Man kann dieses Hindernis umgehen , indem man die fertigen Bilder im Buch 'inrOOO.gsbook' speichert . Die gespeicherten Bilder kann man dann anschließend - mit Hilfe der Prozedur B7 von J. Buschmann (s. /2/) - an einem der Geräte STATOS , XYNETICS , VERSATEC zeichnen lassen . Abb. 7 zeigt ein Bild , das auf diesem Wege - FIGUR - GSBOOK - B7 - entstand . Bei der B7-Prozedur ist darauf zu achten , daß nur Bilder , die man mit dem Befehl ROUT freigibt , auf ein neues Blatt kommen . OUT hingegen zeichnet das Bild auf ein bereits "benutztes" Blatt Papier! (s. auch /2/) . Eine stark vereinfachte Version der B7-Prozedur , BRAUS , wird im Kap. 11 abgedruckt.

Die Prozedur FIGUR

entspricht im wesentlichen der Stapel-Prozedur FIGURB . Sie benötigt aber eine zusätzliche Datei , 'tsoOOO.MEMORY(LAFIGU)' , um die - erfahrungsgemäß mühsam erarbeiteten - Daten des Bilder-Rahmens und die Rahmen-Beschriftungen speichern zu können . Diese Datei enthält in der Regel die letzte Version der schon besprochenen NAMELIST LAFIGU (s. 43). Anders als FIGURB zeichnet FIGUR immer auf DIN A4-Querformat Blätter. Es ist auch bei FIGUR möglich , mit Hilfe der Dateien MEMORY(ABB1) - ABB4) vorgefertigte Bild-Texte in Bereitschaft zu halten.

Prozedur-Aufruf : EX 'tso000.SERVUS.CLIST(FIGUR)' .

In der ersten Prozedur-Stufe allokiert FIGUR (die Prozedurliste ist im Kap. 11 abgedruckt) zunächst die benötigten Dateien . Die hierfür erforderlichen Angaben fragt FIGUR vom Benutzer ab (s. auch die Bemerkungen unter 11.1).
In der zweiten Prozedur-Stufe beginnt ein FORTRAN77-Dialog . Hier wird der Benutzer nach den gewünschten Funktionen abgefragt (mit LOS ist die Ordnungszahl der Funktion gemeint) , sowie nach den Daten , die zur Rahmen-Herstellung (s. ABB. 6) und für die Beschriftungen notwendig sind .
Mit den Funktions-Nummern wird , wie in FIGURB verfahren: eine positive Zahl , LOS > 0 bedeutet , daß mit den LOS-ten Funktion ein neues Bild beginnt , eine negative Zahl bewirkt dagegen , daß die (-LOS)-te Funktion noch in das angefangene Bild kommt .
Der Benutzer kann die Kurven eines Bildes einzeln aus der Datei zusammensuchen , in diesem Falle muß er beim Befragen JEDE = 0 setzen. Er kann aber - sofern die Funktionen in der Datei den gleichen "Abstand" zueinander haben - ein halbautomatisches Suchverfahren benutzen : wenn er z.B. JEDE = 6 setzt , dann werden die Funktionen LOS , LOS+6 , LOS+12 ,... aus der Datei ausgesucht bis zum erreichen der Ordnungszahl LETZTE . Danach kann die Suche manuell weitergehen . Falls man JEDE negativ setzt , dann geht das Suchen von den hohen zu den niedrigen Ordnungszahlen : LOS , LOS-6, LOS-12, usw.
Jedesmal , wenn eine Funktion bereit steht , wird ihre Namenszeile ausgedruckt (wie auf der Abb. 1) und der Benutzer um Zustimmung gebeten . Hier besteht die Möglichkeit , falls die Funktion - verglichen mit den anderen Funktionen - zu klein oder zu groß ist - mit FAKTOR NACHPRUEFEN zu antworten . Bei dieser Antwort wird vom Benutzer ein Faktor erfragt , mit dem die Prozedur dann die Ordinaten multipliziert .
Die Achsen- und die Funktions- Beschriftungen sowie die Bildunterschrift kann man - anhand der Funktions-Kenngrößen - selber schreiben und diese Texte dann für mehrmalige Anwendung speichern .
Mit der erfragten Eingabe wird dann das Bild fertiggestellt und am GA gezeigt . Falls erwünscht , kann man in einem zweiten Durchlauf das gezeigte Bild speichern . Man kann aber auch nur den Bild-Rahmen und/oder die Beschriftungen ändern , oder mit der Suche von neuen

Funktionen für ein neues Bild beginnen .
Die FORTRAN-Prozedur FIGUR geht dann solange weiter, bis der Benutzer sie mit einer Funktions-Ordnungszahl LOS = 0 oder mit PROGRAMMSTATUS < 0 stoppt .

Die FIGUR - Bilder sind wie folgt zerteilt:

FIGUR00j.RAHMEN
FIGUR00j.INHALT.XTEXT
FIGUR00j.INHALT.FTEXT
FIGUR00j.INHALT.KURVEN
FIGUR00j.INHALT.KURVTX
FIGUR00j.TEXT

j steht hier für die laufende Nummer des Bildes. Anstatt FIGUR00j kann der Benutzer eine andere - GS-zulässige - Name einsetzen.
Die Verteilung der zu speichernden Bildelemente in diese Teilbilder erfolgt genau so, wie dies bei FIGURB der Fall war.

10. Funktionen aus einer SERVUS-Datei werden als eine Funktions--Oberflache gezeichnet.

10.1 Das Prozedurpaar FIGUR/FIGURB kann mehrere Elemente einer Funktions-Schar gleichzeitig in einer Bild-Ebene zeigen . Die Prozeduren DDFIG/DDFIGB dienen dazu , "verwandte" Funktionen als eine dreidimensionale Oberfläche abzubilden. Sie stellen zu diesem Zweck die Funktionen in parallelen Ebenen nebeneinander und zeichnen von der so entstandenen Funktions-Oberfläche - mit Hilfe der DDPLO1 Routine /5/ - ein perspektivisches Bild . Die fertigen Bilder können genau so , wie dies bei den FIGUR-Prozeduren der Fall war , gespeichert werden (s. 9.1 , S. 40).
Als Vergleichs-Beispiel zeigen die Abb. 7 und 8 dieselben Funktionen, beim ersten Male mit FIGUR und beim zweiten Male mit DDFIG dargestellt.

Der Aufbau eines Oberflächen-Bildes unterscheidet sich von der Kurvenschar-Darstellung. Es gibt z.B. zwar auch hier einen inneren Rahmen , er ist aber unsichtbar und ist nur da , um das Oberflächen-Bild in die Mitte des Blattes einzuordnen. Auch die Markierung und die Beschriftung der einzelnen Funktionen entfällt bei diesen 3D-Prozeduren.
Die Oberflächen-Bilder sind dafür vielschichtiger: die Fläche selbst besteht aus der X=konst.- und aus der Z=konst.-Linien (im Folgenden ist Z immer der Scharen-Parameter der gezeigten Kurven , s. auch Abb. 9). Dazu können noch folgende Seh-Stützen kommen (s. Abb. 8) :

- die seitlichen Begrenzungsflächen,
- ein Richtungspfeil mit Beschriftung (17 Zeichen) je Achse,
- ein - die Oberfläche umschließender - "Koordinatenwürfel", bestehend aus ausgezogenen oder aus gestrichelten Linien,
- eine Bemaßung der Achsen dieses Würfels durch Markierungen.

10.2 S t a p e l - V e r f a h r e n f ü r d i e O b e r f l ä c h e n - D a r s t e l l u n g.

Die Prozedur DDFIGB erzeugt Oberflächen-Bilder von der gespeicherten Funktionen im Stapelbetrieb. In einem Auftrag kann man bis zu 100 Oberflächen abbilden , die dann zu einem Gesamtbild zusammengelegt und dieses dann durch das Xynetics- , Versatec- oder Statos-Zeichengerät ausgegeben wird. Das Gesamtbild wird auch hier als "T001" gespeichert.
Die einzelnen Teilbilder von T001 haben folgende Aufbau:

```
T001.RAHMEN0j
T001.INHALT0j.KONSX0j
T001.INHALT0j.KONSZ0j    ,   0j = 01 , ... , 99 , 00
T001.INHALT0j.LINKS
T001.INHALT0j.RECHTS
T001.INHALT0j.HOCH
T001.INHALT0j.KOORD0j
T001.TEXT0j      .
```

Das Teilbild .RAHMEN0j enthält nur den äußeren Rahmen (Bild--Trennnetz), in den Teil .TEXT0j kommt die Bildunterschrift , alles andere wird im Teil .INHALT0j. gespeichert. Die X=konst. bzw. Z=konst. Linien kommen in die Teilbilder .KONSX0j bzw. .KONSZ0j , die Pfeile , nebst Achsen-Beschriftungen kommen in die drei Teile .LINKS , .RECHTS bzw. .HOCH und der Koordinaten-Würfel in den Teil .KOORD0j .

Der Benutzer kann auch hier das Bild nach eigenem Geschmack aufbauen - was u.U. umfangreiche Eingabe erfordert , er kann aber auch die wenig Arbeit verlangenden Standard-Darstellungen verwenden.

In der Regel (Mindesteingabe) wird nur die "nackte" Oberfläche gezeigt und zwar in Parallelprojektion , mit Kipp- und Drehwinkel von 75 bzw. 300 Grad . Das Bild ist unverzerrt, d.h. alle Raumrichtungen werden gleich behandelt. Im Regelfall werden die einzelnen Teilbilder im T001 mit 1 beginnend durchnumeriert . Sie erhalten keine Bild--Unterschrift außer der Bildnummer. Die "Abb. 1" im Bild Abb. 9 zeigt eine Fläche mit Mindesteingabe .
Es besteht aber - wie bei FIGURB - auch die Möglichkeit , Bildnummer,

Achsen-Beschriftungen und Bild-Unterschrift vorzugeben . Der Benutzer kann den für die Abbildung günstigsten Sichtpunkt aussuchen . Er kann auch die gespiegelte oder die "komplementäre" Version seiner Fläche zeichnen lassen (vgl. Abb. 10) , usw.

Die Liste der Hintergrund-Prozedur DDFIGB (mit dem VERSATEC-Gerät):

Name : tso000.SERVUS.CNTL(DDFIGB)

```
//inr000d  JOB (0abc,xyz,p9x9y),benutzer,MSGLEVEL=(1,1),REGION=2500K
//A EXEC F7CLG,PLOT=GS7,USER='SYS2.TRACEGS7'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C     D D F I G B  ( STAPELVERSION VON DDFIG )
      CHARACTER*4 INST
      DATA IRE/15/,IDIN/6/,NETZ/1/,MEM/30/,KPRI/2/,INST/'INR '/
      CALL DDFIGB(IRE,IDIN,NETZ,MEM,KPRI,INST)
      STOP
      END
//*
//L.BIBS DD DSN=INR105.SERVUS.LOAD,DISP=SHR
//L.SYSIN DD *
 INCLUDE BIBS(SERDIO,DDFIGB,NEXT,PAPIER,WUERFL,FENST4,NARS)
 ENTRY MAIN
//*
//G.PLOTPARM DD *
 &PLOT XMAX=290.1,YMAX=23., &END
//G.COMM     DD SYSOUT=*
//G.FT15F001 DD DSN=inr000.vielbild.DATA,DISP=(OLD,KEEP)
//G.FT31F001 DD DSN=tso000.MEMORY(ABB1),DISP=(OLD,KEEP)
//G.FT32F001 DD DSN=tso000.MEMORY(ABB2),DISP=(OLD,KEEP)
//G.FT33F001 DD DSN=tso000.MEMORY(ABB3),DISP=(OLD,KEEP)
//G.FT34F001 DD DSN=tso000.MEMORY(ABB4),DISP=(OLD,KEEP)
//G.TRACEGS7 DD DSN=inr000.gsbook,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LADDFI INLIST=-6,LAUF=18,18, &END
 &LADDFI INLIST=-11,NABB=7,ABTEX(1:8)='ZWEITENS',ISCH=1,BADH=0.6,IMO=5,
 FMIST=-150.,FMAST=150.,FEDST=50., &END
 ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
 &LADDFI ABTEX(1:9)='SECHSTENS',ABSTND=0.,ITX=1,
 TEXTX='KANALHOEHE ( CM )',TEXTF='GESCHW.( CM/SEC )',
 TEXTZ='ZEIT  ( MILISEC )', &END
 &LADDFI ABTEX(1:9)='SIEBTENS ',ITX=34,ISCH=42,FRAR$=6., &END
 ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
 &LADDFI LAUF(1)=0, &END
//*
//P EXEC SVPLOT,SPACE=50
//
```

Die gezeigte Eingabe entspricht dem Bild Abb. 9.

Eingabebeschreibung:

Die Eingabe besteht aus allgemeinen Angaben zum Zeichnungs-Verfahren und aus speziellen Anweisungen für jedes einzelne Bild.
Die allgemeinen Verfahrens-Hinweise sind im Aufruf-Kopf der Prozedur enthalten und man kann sie in der DATA-Anweisung ändern . Diese Steuergrößen sind:

(IRE,IDIN,NETZ,MEM,KPRI,INST).

Von diesen Größen ist nur INST ein CHARACTER*4 Wort , alle anderen sind INTEGERs . Sie haben die folgende Bedeutung:

IRE	ist die symbolische Nummer der SERVUS-Datei,
IDIN	ist die DIN-Größe des Bildes ,
> 0	zeichnet das Bild in DIN-A-"IDIN"-Querformat und
< 0	in DIN-A-"-IDIN"-Hochformat.
NETZ > 0	erzeugt ein Trennungsnetz zwischen den Bildern. Falls
2 | -2	ist und ABS(IDIN) < 5 , dann erhält jedes Teilbild das KFK-Emblem (s. z.B. Abb. 8).
MEM	wählt den Zeichengerät: für
1 | 10	wird der STATOS-Plotter benutzt , für
2 | 20	der XYNETICS-Plotter und für
3 | 30	der VERSATEC-Plotter . Falls
MEM =	
10 | 20 | 30	ist , dann wird das Gesamtbild T001 auch gespeichert.
KPRI	ist die Ausdruck-Intensität der Begleit-Druckausgabe.
INST	4 Zeichen , die auf dem KfK-Emblem mit erscheinen (Institutskürzel , +INR + auf der Abb. 8).

Die speziellen Bild-Informationen werden im NAMELIST-Format eingegeben , für jedes Teil-Bild eine Liste "LADDFI" :

```
/LADDFI/ INLIST,LAUF,XMIST,XMAST,XEDST,FMIST,FMAST,FEDST,
         SCR$,XABB$,FABB$,XAUF$,FAUF$,XRAR$,FRAR$,TAP$,TAT$,TAZ$,
         BADH,ABSTND,THETA,PHI,KZ,ISCH,IRIS,IMO,MRE,
         NABB,ITX,ABTEX,TEXTX,TEXTF,TEXTZ
```

Von den Listen-Größen ist ABTEX ein CHARACTER*70 Name , TEXTX , TEXTZ und TEXTF sind CHARACTER*17 Namen , KZ,ISCH,IRIS,IMO,MRE,NABB, INLIST und ITX sind INTEGERs und LAUF ist ein 50-er Feld von INTEGERs . Alle anderen Daten sind REAL*4 Zahlen .

Die Bedeutung der einzelnen Listen-Größen:

INLIST , LAUF dürfen auch bei Mindesteingabe nicht fehlen. Sie bestimmen zusammen die Zahlenmenge der zu zeichnenden Funktionen. Falls

INLIST > 0 ist , besteht sie aus den Zahlen
{ LAUF(1),LAUF(2),...,LAUF(INLIST) } .
Bei dieser Eingabe kommt nach der ersten Funktion eine weitere nur dann in das angefangene Bild , falls das zugehörige LAUF(j) negativ ist . Ein Index LAUF(k) ≥ 0 beendet das Bild . Falls

INLIST = 0 ist , ist die Zahlenmenge die folgende :
{ LAUF(1) ≤ i ≤ LAUF(2) } bzw.
{ LAUF(1) ≥ i ≥ LAUF(2) } und falls

INLIST < 0 ist , hat man die -INLIST Ordnungszahlen :
{ LAUF(1),[LAUF(1)+LAUF(2)],[LAUF(1)+2*LAUF(2)],... }.
Bei den Eingabe-Typen INLIST = 0 und INLIST < 0 werden - im Gegensatz zu dem "INLIST > 0"-Fall - alle Funktionen , die durch die Zahlenmenge bestimmt sind , in dasselbe Bild eingezeichnet .

LAUF(1) = 0 : eine Eingabe-Liste mit LAUF(1) = 0 stoppt die Prozedur DDFIGB .

Die folgenden Daten umreißen das - für das Bild vorsehene - Datenfenster (s. auch Kap. 2.3 und 6.2):

XMIST,XMAST bestimmen den erlaubten Werte-Bereich auf der X-Achse,
FMIST,FMAST den entsprechenden Bereich auf der F-Achse.
XEDST,FEDST sind die vorgegebenen Skalenlängen (s. S. 11).

Falls bei einer Eingabe XMIST ≥ XMAST ist , oder XEDST = 0 , dann benutzt die Prozedur anstelle dieser Größen XMI , XMA und XED von der ersten Funktion des Bildes. Falls auch diese Größen ungeeignet sind , dann werden diese Daten anhand der Abszissenmenge von der Prozedur ermittelt . Entsprechendes gilt bei der Größen FMIST , FMAST und FEDST .

Die folgenden Daten bestimmen den inneren Rahmen und die Schrift :

SCR$ ist die Höhe der verwendeten Schriftzeichen,

XAUF$, FAUF$ sind die Koordinaten der linken , unteren Ecke des unsichtbaren , inneren Rahmens und

XRAR$, FRAR$ sind die Abstände zum Bildrand am rechten , bzw. am oberen Ende des inneren Rahmens.

XABB$, FABB$ sind die Anfangs-Koordinaten der Bildunterschrift.

Die Bedeutung der obigen Größen kann man auch aus der Abb. 6 entnehmen.

Alle hier beschriebenen Größen SCR$,XAUF$ - ... - FABB$ sind in den relativen Einheiten UNITX bzw. UNITF anzugeben : UNITX ist 1/32. der X-Ausdehnung des Bildes , UNITF ist 1/32. der F-Ausdehnung .

Eingabegrößen für die Richtungspfeile (in Schrifthöhe-Einheiten):

TAP$ ist der Abstand des Richtungpfeiles zur jeweiligen Achse und

TAT$ ist der entsprechende Abstand der Achsen-Beschriftung.

TAZ$ bestimmt die Größe der Markierungen auf dem Koordinaten-Würfel.

Die folgenden Daten beziehen sich auf die Abbildung und auf die Art der Flächen-Darstellung:

ABSTND ist der Abstand des Betrachters von der Fläche in cm,

≤ 0 bedeutet Parallelprojektion.

THETA , PHI in Grad bestimmen der Standpunkt des Betrachters:

THETA ist der Kipp-(Höhen-)-Winkel des Sichtpunktes und

PHI ist der Dreh-(Azimutal-)-Winkel.

BADH > 0 ist die Höhe/Breite-Verhältnis im Bild. Beim

BADH $= 1.$ entsteht ein unverzerrtes Bild .

KZ $= 1$ zeichnet nur die nackte Oberfläche , bei

KZ $= 2$ sind auch die seitlichen Begrennzungsflächen sichtbar.

IRIS steuert die Spiegelungen der Funktionsfläche :

$= 0$ liefert die ursprüngliche ,

$= 1$ ergibt die komplementäre und

$= 2$ die gespiegelte Fläche (s. Abb 10).

ISCH steuert die zusätzliche Sichtunterstützungen: bei

$= 0$ wird nur die Fläche gezeichnet, bei

$= 1$ werden auch die Richtungspfeile samt Beschriftungen

gezeigt und bei

= 2 werden auch die Markierungen auf die Achsen gezeichnet.

ISCH ≥ 3 zeigt auch den Koordinatenwürfel :

= 3 zeichnet einen Würfel mit ausgezogenen Kanten

> 3 ergibt einen gestrichelten Würfel und zwar

ISCH = j zeigt j-2 Striche je Kante.

IMO,MRE setzen Markierungen auf die Z-Achse: im Falle ISCH > 1 werden alle IMO-ten Kurven - mit den MRE-ten beginnend - markiert.

Die restlichen Daten beziehen sich auf die Beschriftungen und auf die Kurvendarstellung:

NABB ist die vorgesehene Bild-Nummer . Falls

= 0 ist , werden die Bilder durchnumeriert .

ITX entscheidet über die Beschriftung :

ITX = 0 erzeugt ein Bild , bei der nur ABTEX und die Funktionsgrößen NAMX(1),MASX(1) und NAMF(1),MASF(1) zum Beschriften benutzt werden. Über der Z-Achse steht jetzt "SCHAREN-PARAMETER" (Abb. 9 , "ZWEITENS"). Bei

ITX > 0 werden die Achsen mit den Texteingaben TEXTX , TEXTZ bzw. TEXTF beschriftet. Falls hierbei

ITX = 30+j ist, mit 0 < j < 5 , dann kommt die Bildunterschrift aus der Datei MEMORY(ABBj) , ansonsten wird wieder ABTEX benutzt.

ABTEX enthält die Bildunterschrift (≤ 70 Zeichen),

TEXTX , TEXTZ und TEXTF enthalten die Pfeil-Beschriftungen für die X- , Z- und F-Achse (≤ 17 Zeichen).

10.3 Oberflächen-Darstellung im Dialog.

Die Prozedur DDFIG

entspricht im wesentlichen der Stapel-Prozedur DDFIGB . Hier wird auch eine zusätzliche Datei , 'tso000.MEMORY(LADDFI)' benötigt , um

die Daten des Rahmens , des Sicht-Punktes und der Rahmen-Beschriftungen speichern zu können . Diese Daten werden in MEMORY(LADDFI) in der Form einer NAMELIST gespeichert , die mit der ab S. 53 besprochenen Liste - LADDFI identisch ist. Auch DDFIG zeichnet immer auf DIN A4-Querformat-Blätter und man kann auch hier die Dateien MEMORY(ABB1) - ABB4) benutzen , um im Laufe der Prozedur nach vorgefertigten Unterschriften zu greifen.

Prozedur-Aufruf : EX 'tso000.SERVUS.CLIST(DDFIG)' .

Die Prozedur DDFIG läuft in etwa so ab wie FIGUR . Unterschiede gibt es in einigen Rahmen-Eingaben : FIGUR verlangt mehr Beschriftungen , bei DDFIG muß man dafür Projektions-Daten , wie Projektions-Abstand, Kipp- und Drehwinkel sowie Darstellungs-Kenngrößen eingeben .
Für die Einzelheiten des Dialog-Ablaufes möge man beim FIGUR (Kap. 9.3) nachschauen .

Wichtig: die Prozedur DDFIG braucht sehr viel Speicherraum . Schon eine Fläche von 50•50 Punkten kann 200 K Platz beanspruchen. Falls man das Bild auch noch speichert , braucht man den doppelten Raum . Beim Einstieg in die TSO-Sitzung ist also ein Space-Parameter S(2000) sehr zu empfehlen. Andernfalls bricht DDFIG beim ersten aufwendigen Bild unvermutet die Arbeit mit einer Fehlermeldung ab.

11. Die SERVUS-Kommandoprozeduren.

11.1 Über Dialoge im Kommando- und in FORTRAN-Mode.

Beide Dialog-Verfahren erfragen der Reihe nach vom Benutzer die für die jeweilige Prozedur benötigten Eingabedaten.
Alle im SERVUS-System zu Wort kommenden Dialoge zeigen - vor einer Eingabe-Forderung die zuletzt benutzte Werte der fraglichen Größe am Bildschirm in folgender Form : EINGABE / letzte Wert / =: .
Mehrere "=:" -Zeichen bedeuten entsprechend viele Eingabe-Größen .

Leider muß man sich daran gewöhnen, daß sich die Regeln des FORTRAN--Dialogs (s. /6/) von der Regeln des Kommando-Dialogs in einigen Punkten wesentlich und verwirrend unterscheiden :

- Jedes Mal , wenn die FORTRAN-Prozedur vom Benutzer eine Eingabe erwartet , erscheint am Bildschirm ein "?" . Der Kommando-Dialog hat keine besondere Zeichen zum Eingabe-Fordern.
- im FORTRAN Dialog werden alle Zeichen der Tastatur akkzeptiert. Im gegensatz dazu sind beim Kommando-Dialog die folgenden Zeichen Tabu: + - * / = > < | ¬ & (, '
  Die erste 10 dieser Zeichen führen zum sofortigen Abbruch der Prozedur, die letzte 3 werden als Leerzeichen mißdeutet. Besonders mißlich ist diese Geschichte beim Zeichen "/" : es ist grade dieses Zeichen , das man erfahrungsgemäß bei einem FORTRAN-Dialog sehr häufig benutzt, um mehrere Eingabegrößen mit "einem Wort" zu bestätigen (s. u.).
- Beim Kommando-Dialog bedeutet eine leere Eingabe (einfaches Drücken auf die "Enter"-Taste) ein "ja" , bzw. daß die fragliche Größe ihren bisherigen Wert behält . Von der FORTRAN--Prozedur wird eine leere Eingabe nicht akzeptiert (sie reagiert mit einen "?") . Falls man den bisherigen Wert der Größe beibehalten möchte , tippt man eine Komma "," ein . Wenn man mehrere geforderte Größen unverändert stehen lassen will , kann man das mit einem "/" erreichen .
- Textvariablen nimmt der Kommando-Dialog ohne Hochkommata

(TEXTTEXT) , "'" wird hier als Leerzeichen angesehen. Der FORTRAN-Dialog nimmt sie nur mit Hochkommata ('TEXTTEXT').

- Bei einer fehlerhaften Eingabe (Zeichen statt Ziffer z.B.) wiederholt der FORTRAN-Dialog die ganze Frage (wichtig bei Fragen , bei denen die Prozedur eine Antwort aus mehreren Zahlen/Wörtern erwartet). Der Kommando-Dialog fragt nie nach.

11.2 Die Kommandoprozedur IDA.

Name : tso000.SERVUS.CLIST(IDA)

```
PROC 0  USER(inr000) DATEI(test) VOL(bat00c) NORM(fol8)
CONTROL MAIN MSG PROMPT
WRITE
WRITE PROZEDUR       IDA
WRITE KOPFZEILE EINER SERVUS DATEI WIRD (NEU-) GESCHRIEBEN
WRITE
SET &B=&STR( )
WRITE ANGABEN ZU DER DATEI :
WRITENR PROJEKT / &USER / =: ( "ENTER"=JA )
     READ &US
IF &US=&STR() THEN  GOTO E1
SET &USER = &US
E1: WRITENR NAME / &DATEI / =: ( "ENTER"=JA )
     READ &DAT
IF &DAT=&STR() THEN  GOTO E2
IF &LENGTH(&DAT) > 8 THEN SET &DAT = &SUBSTR(1:8,&DAT)
SET &DATEI = &DAT
E2: WRITENR NORM / &NORM / =: +
  ( "ENTER"(=JA) | GRA4 | GRA8 | FOL8 | PLOT )
     READ &NOR
IF &NOR=&STR() THEN  GOTO E3
SET &NORM=&NOR
E3: SET &SYSDVAL = &DATEI&B&NORM&B&USER&B&VOL&STR( END)
WRITENR IST DIE DATEI +
        '&USER..&DATEI..DATA' ALLOKIERT =: ( "ENTER"=JA )
     READ &ANS
IF &ANS = &STR()  THEN GOTO A5
ATTRIB SERVAT BLKSIZE(19069) LRECL(X) DSORG(PS) RECFM(V B S)
WRITENR PLATTE / &VOL / =: ( "ENTER"=JA )
     READ &ANS
IF &ANS=&STR() THEN GOTO A1
SET &VOL=&ANS
A1: ALLOC DA('&USER..&DATEI..DATA') NEW VOLUME(&VOL) UNIT(DISK) +
    SPACE(10,2) CYL RELEASE  USING(SERVAT)
FREE ATTRLIST(SERVAT)
A5: WRITE DIE DATEI IST ALLOKIERT .  (RE-)NORMIERUNG DER DATEI:
ALLOC F(FT15F001) DA('&USER..&DATEI..DATA')
CALL 'INR105.SERVUS.LOAD(IDAJOB)' '&SYSDVAL  '
FREE F(FT15F001)
WRITE ENDE DER PROZEDUR IDA
EXIT
```

11.3 D i e K o m m a n d o p r o z e d u r E N D E .

Diese Prozedur braucht neben der zu reparierenden Datei
die Gedächtnisstütze MEMORY(DATAS) .

Name : tso000.SERVUS.CLIST(ENDE)

```
PROC 0 AL(0)
CONTROL MAIN MSG PROMPT
WRITE
WRITE PROZEDUR      ENDE
WRITE EINE SERVUS-DATEI WIRD MIT EINER ENDZEILE VERSEHEN
WRITE
/* EINGBEDATEN HOLEN UND UEBERPRUEFEN
SET &B = &STR( )
ALLOC FILE(OUTDA) DA('tso000.MEMORY(DATAS)')
OPENFILE OUTDA
GETFILE OUTDA
SET &SYSDVAL=&OUTDA
CLOSFILE OUTDA
READDVAL &USER &DATEI1 &DATEI2 &DATEI3  &DATREG
M1: WRITE DATEI NR "15" / '&USER..&DATEI1..DATA' / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = DIE DATEI "15" WIRD NICHT ALLOKIERT
     READ &ANS
IF &ANS=&STR() THEN GOTO M10
IF &ANS=&STR(PROJEKT) | &ANS=&STR(PR) THEN GOTO M6
IF &ANS¬=&STR(NAME) && &ANS¬= &STR(NM) THEN GOTO M99
WRITENR NAME / &DATEI1 / =:
     READ &DATEI1
GOTO M1
M6: WRITENR PROJEKT / &USER / =:
     READ &USER
GOTO M1
M10: SET &AL=1
SET &SYSDVAL = &USER&B&DATEI1&B&DATEI2&B&DATEI3&B&DATREG&STR( END)
SET &L = &LENGTH(&SYSDVAL)
OPENFILE OUTDA OUTPUT
SET &OUTDA=&SUBSTR(1:&L,&SYSDVAL)
PUTFILE OUTDA
CLOSFILE OUTDA
FREE F(OUTDA)
WRITE BEGINN DER PROZEDUR
ALLOC DA('&USER..&DATEI1..DATA') F(FT15F001)
CALL 'INR105.SERVUS.LOAD(ENDE)'
FREE F(FT15F001)
M99: WRITE ENDE DER PROZEDUR "ENDE"
EXIT
```

11.4 Die Kommandoprozedur DATAS.

DATAS braucht mindestens eine und höchstens vier SERVUS-Dateien.
memory(DATAS) speichert die Prozedur-Eingaben.

Name : tso000.SERVUS.CLIST(DATAS)

```
PROC 0 AL(0) AR(0)
CONTROL MAIN MSG PROMPT
WRITE
WRITE PROZEDUR      DATAS
WRITE EINE SERVUS-DATEI WIRD GESICHTET ( UND KOPIERT )
WRITE
/* EINGBEDATEN HOLEN UND UEBERPRUEFEN
SET &B = &STR( )
ALLOC FILE(OUTDA) DA('tso000.MEMORY(DATAS)')
OPENFILE OUTDA
GETFILE OUTDA
SET &SYSDVAL=&OUTDA
CLOSFILE OUTDA
READDVAL &USER &DATEI1 &DATEI2 &DATEI3  &DATREG
WRITE
M1: WRITE LESEDATEI NR "15" / '&USER..&DATEI1..DATA' / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = DIE DATEI "15" WIRD NICHT ALLOKIERT
     READ &ANS
IF &ANS=&STR( ) THEN GOTO M10
IF &ANS=&STR(PROJEKT) | &ANS=&STR(PR) THEN GOTO M6
IF &ANS¬=&STR(NAME) && &ANS¬= &STR(NM) THEN GOTO M41
WRITENR NAME / &DATEI1 / =:
     READ &DATEI1
GOTO M1
M6: WRITENR PROJEKT / &USER / =:
     READ &USER
GOTO M1
M10: SET &AL=1
WRITE
M11: WRITE LESEDATEI NR "16" / '&USER..&DATEI2..DATA' / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = DIE DATEI "16" WIRD NICHT ALLOKIERT
     READ &ANS
IF &ANS=&STR( ) THEN GOTO M20
IF &ANS=&STR(PROJEKT) | &ANS=&STR(PR) THEN GOTO M16
IF &ANS¬=&STR(NAME) && &ANS¬= &STR(NM) THEN GOTO M31
WRITENR NAME / &DATEI2 / =:
     READ &DATEI2
GOTO M11
M16: WRITENR PROJEKT / &USER / =:
     READ &USER
GOTO M11
M20: SET &AL=2
WRITE
M21: WRITE LESEDATEI NR "17" / '&USER..&DATEI3..DATA' / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
```

```
WRITENR SONSTIGES = DIE DATEI "17" WIRD NICHT ALLOKIERT
     READ &ANS
IF &ANS=&STR( ) THEN GOTO M30
IF &ANS=&STR(PROJEKT) | &ANS=&STR(PR) THEN GOTO M26
IF &ANS¬=&STR(NAME) && &ANS¬= &STR(NM) THEN GOTO M31
WRITENR NAME / &DATEI3 / =:
     READ &DATEI3
GOTO M21
M26: WRITENR PROJEKT / &USER / =:
     READ &USER
GOTO M21
M30: SET &AL=3
WRITE
M31: WRITE SCHREIB-DATEI "26" / '&USER..&DATREG..DATA' / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = DIE DATEI "26" WIRD NICHT ALLOKIERT
     READ &ANS
IF &ANS=&STR( ) THEN GOTO M40
IF &ANS=&STR(PROJEKT) | &ANS=&STR(PR) THEN GOTO M36
IF &ANS¬=&STR(NAME) && &ANS¬= &STR(NM) THEN GOTO M41
WRITENR NAME / &DATREG / =:
     READ &DATREG
GOTO M31
M36: WRITENR PROJEKT / &USER / =:
     READ &USER
GOTO M31
M40: SET &AR=1
WRITE
M41: SET &SYSDVAL = &USER&B&DATEI1&B&DATEI2&B&DATEI3&B&DATREG&STR( END)
SET &L = &LENGTH(&SYSDVAL)
OPENFILE OUTDA OUTPUT
SET &OUTDA=&SUBSTR(1:&L,&SYSDVAL)
PUTFILE OUTDA
CLOSFILE OUTDA
FREE F(OUTDA)
WRITE BEGINN DER PROZEDUR
IF &AL<=0 THEN  GOTO M99
IF &AL>0 THEN  ALLOC DA('&USER..&DATEI1..DATA') F(FT15F001)
IF &AL>1 THEN  ALLOC DA('&USER..&DATEI2..DATA') F(FT16F001)
IF &AL>2 THEN  ALLOC DA('&USER..&DATEI3..DATA') F(FT17F001)
IF &AR>0 THEN  ALLOC DA('&USER..&DATREG..DATA') F(FT26F001)
CALL 'INR105.SERVUS.LOAD(DATAS)'
FREE F(FT15F001)
IF &AL<=1 THEN GOTO M48
FREE F(FT16F001)
IF &AL<=2 THEN GOTO M48
FREE F(FT17F001)
M48: IF &AR<=0 THEN GOTO M99
FREE F(FT26F001)
M99: WRITE ENDE DER PROZEDUR DATAS
EXIT
```

11.5 Die Kommandoprozedur FIGUR.

FIGUR braucht eine SERVUS-Datei mit der Schreib-Norm "GRA4".
Die Prozedur-Eingaben kommen aus MEMORY(FIGUR).

Name : tso000.SERVUS.CLIST(FIGUR)

```
PROC 0  AL(0) AB(0)
CONTROL MAIN MSG PROMPT
SET &B = &STR( )
WRITE
WRITE PROZEDUR       FIGUR
WRITE FUNKTIONEN / FUNKTIONSSCHAREN  WERDEN GEZEICHNET.
WRITE
/* EINGBEDATEN HOLEN UND UEBERPRUEFEN
ALLOC FILE(OUTDA) DA('tso000.MEMORY(FIGUR)')
OPENFILE OUTDA
GETFILE OUTDA
SET &SYSDVAL=&OUTDA
CLOSFILE OUTDA
READDVAL &USER &DATEI &USBU &BUCH
M1: WRITE FUNKTIONS-DATEI - "15" / &USER..&DATEI..DATA / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = DIE DATEI "15" WIRD NICHT ALLOKIERT
     READ  &ANS
IF &ANS=&STR() THEN GOTO M10
IF &ANS=&STR(PROJEKT) |  &ANS=&STR(PR)   THEN GOTO M6
IF &ANS¬=&STR(NAME) &&  &ANS¬=&STR(NM)  THEN GOTO M99
WRITENR NAME / &DATEI / =:
     READ &DATEI
GOTO M1
M6: WRITENR PROJEKT / &USER / =:
     READ &USER
GOTO M1
M10: WRITE "BUCH" ZUM BILDSPEICHERN  / &USBU..&BUCH / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = KEIN "BUCH" WIRD ALLOKIERT
     READ  &ANS
IF &ANS=&STR() THEN GOTO M19
IF &ANS=&STR(PROJEKT) |  &ANS=&STR(PR)  THEN GOTO M16
IF &ANS¬=&STR(NAME) &&  &ANS¬=&STR(NM)  THEN GOTO M20
WRITENR NAME / &BUCH / =:
     READ &BUCH
GOTO M10
M16: WRITENR PROJEKT / &USBU / =:
     READ &USBU
GOTO M10
M19: SET &AB=1
M20: SET &SYSDVAL = &USER&B&DATEI&B&USBU&B&BUCH&STR( END)
SET &L = &LENGTH(&SYSDVAL)
OPENFILE OUTDA OUTPUT
SET &OUTDA=&SUBSTR(1:&L,&SYSDVAL)
PUTFILE OUTDA
CLOSFILE OUTDA
```

```
FREE F(OUTDA)
WRITE BEGINN DER PROZEDUR
ALLOC DA(*) F(COMM)
ALLOC DA('&USER..&DATEI..DATA') F(FT15F001)
ALLOC DA('tso000.MEMORY(LAFIGU)') F(FT10F001)
 ALLOC DA('tso000.MEMORY(ABB1)') F(FT31F001)
 ALLOC DA('tso000.MEMORY(ABB2)') F(FT32F001)
 ALLOC DA('tso000.MEMORY(ABB3)') F(FT33F001)
 ALLOC DA('tso000.MEMORY(ABB4)') F(FT34F001)
IF &AB = 1 THEN ALLOC DA('&USBU..&BUCH') F(TRACEGS7)
   GSA GT(GAT)
CALL 'INR105.SERVUS.LOAD(FIGUR)'
FREE F(FT15F001)
FREE F(FT10F001)
FREE F(FT31F001)
FREE F(FT32F001)
FREE F(FT33F001)
FREE F(FT34F001)
IF &AB=1  THEN       FREE F(TRACEGS7)
M99: WRITE ENDE DER PROZEDUR FIGUR
EXIT
```

11.6 Die Kommandoprozedur DDFIG.

DDFIG braucht eine SERVUS-Datei mit der Schreib-Norm "GRA4".
Die Prozedur-Eingaben kommen aus MEMORY(FIGUR).

Name : tso000.SERVUS.CLIST(DDFIG)

```
PROC 0  AL(0) AB(0)
CONTROL MAIN MSG PROMPT
SET &B = &STR( )
WRITE
WRITE PROZEDUR       DDFIG
WRITE DARSTELLUNG VON FUNKTIONEN ALS OBERFLAECHEN .
WRITE
/* EINGBEDATEN HOLEN UND UEBERPRUEFEN
ALLOC FILE(OUTDA) DA('tso000.MEMORY(FIGUR)')
OPENFILE OUTDA
GETFILE OUTDA
SET &SYSDVAL=&OUTDA
CLOSFILE OUTDA
READDVAL &USER &DATEI &USBU &BUCH
M1: WRITE FUNKTIONS-DATEI - "15" / &USER..&DATEI..DATA / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = DIE DATEI "15" WIRD NICHT ALLOKIERT
     READ  &ANS
IF &ANS=&STR() THEN GOTO M10
IF &ANS=&STR(PROJEKT) |  &ANS=&STR(PR)   THEN GOTO M6
IF &ANS¬=&STR(NAME) &&  &ANS¬=&STR(NM)  THEN GOTO M99
WRITENR NAME / &DATEI / =:
     READ &DATEI
GOTO M1
M6: WRITENR PROJEKT / &USER / =:
     READ &USER
GOTO M1
```

```
M10: WRITE "BUCH" ZUM BILDSPEICHERN  / &USBU..&BUCH / =:
WRITE GUELTIGE ANTWORTEN : "ENTER"(=JA) | PROJEKT ( PR ) | NAME ( NM )
WRITENR SONSTIGES = KEIN "BUCH" WIRD ALLOKIERT
     READ  &ANS
IF &ANS=&STR() THEN GOTO M19
IF &ANS=&STR(PROJEKT) |  &ANS=&STR(PR)  THEN GOTO M16
IF &ANS¬=&STR(NAME) &&  &ANS¬=&STR(NM)  THEN GOTO M20
WRITENR NAME / &BUCH / =:
     READ &BUCH
GOTO M10
M16: WRITENR PROJEKT / &USBU / =:
     READ &USBU
GOTO M10
M19: SET &AB=1
M20: SET &SYSDVAL = &USER&B&DATEI&B&USBU&B&BUCH&STR( END)
SET &L = &LENGTH(&SYSDVAL)
OPENFILE OUTDA OUTPUT
SET &OUTDA=&SUBSTR(1:&L,&SYSDVAL)
PUTFILE OUTDA
CLOSFILE OUTDA
FREE F(OUTDA)
WRITE BEGINN DER PROZEDUR
ALLOC DA(*) F(COMM)
ALLOC DA('&USER..&DATEI..DATA') F(FT15F001)
ALLOC DA('tso000.MEMORY(LADDFI)') F(FT10F001)
 ALLOC DA('tso000.MEMORY(ABB1)') F(FT31F001)
 ALLOC DA('tso000.MEMORY(ABB2)') F(FT32F001)
 ALLOC DA('tso000.MEMORY(ABB3)') F(FT33F001)
 ALLOC DA('tso000.MEMORY(ABB4)') F(FT34F001)
IF &AB = 1 THEN ALLOC DA('&USBU..&BUCH') F(TRACEGS7)
   GSA GT(GAT)
CALL 'INR105.SERVUS.LOAD(DDFIG)'
FREE F(FT15F001)
FREE F(FT10F001)
FREE F(FT31F001)
FREE F(FT32F001)
FREE F(FT33F001)
FREE F(FT34F001)
IF &AB=1  THEN       FREE F(TRACEGS7)
M99: WRITE ENDE DER PROZEDUR DDFIG
EXIT
```

11.7 Die Kommandoprozedur BRAUS.

Diese Prozedur benötigt - neben der Buch-Datei - zwei zusätzliche Dateien : MEMORY(BUCH) und SERVUS.CNTL(BRAUSJOB) .

Name : tso000.SERVUS.CLIST(BRAUS)

```
PROC 0 BEN(inr000) XSTAN(50.) NB(0) DIN2(4) DIN3(4) DIN4(4) DIN5(4) +
   BIN2($) BIN3($) BIN4($) BIN5($) FORM2(H) FORM3(H) FORM4(H) FORM5(H) +
   AUS2(OUT) AUS3(OUT) AUS4(OUT) AUS5(OUT) UFA(UFUS) QU(S)
CONTROL MAIN PROMPT MSG
WRITE
WRITE PROCEDUR BRAUS
WRITE BILDER , DIE IN EINEM GS-BUCH  GESPEICHERT SIND
WRITE WERDEN GEZEICHNET.
WRITE
/* EINGBEDATEN HOLEN UND UEBERPRUEFEN
SET &B = &STR( )
ALLOC FILE(OUTDA) DA('tso000.MEMORY(BUCH)')
OPENFILE OUTDA
GETFILE OUTDA
SET &SYSDVAL=&OUTDA
CLOSFILE OUTDA
READDVAL &USER &BUCH &UFUS &UFBU &GER &XFAC &YFAC
WRITENR ZEICHENGERAET / &GER / =: +
    ( "ENTER"=JA | XYN | STA | "SONST"= VER )
READ &ANS
IF &ANS = &STR() THEN SET &ANS=&GER
IF &ANS = &STR(STA) THEN GOTO G8
IF &ANS = &STR(XYN) THEN GOTO G1
SET &GER=&STR(VER)
GOTO G9
G1: WRITE QUALITAET / &QU / =: ( X | Y )
READ &QU
G8: SET &GER=&ANS
G9: WRITE X-FAKTOR / &XFAC / =:
READ &ANS
IF &ANS ¬= &STR() THEN SET &XFAC = &ANS
WRITE Y-FAKTOR / &YFAC / =:
READ &ANS
IF &ANS ¬= &STR() THEN SET &YFAC = &ANS
B1: WRITE BUCH / '&USER..&BUCH' / =:
WRITENR GUELTIGE ANTWORTEN : ENTER(=JA) | NAME (NM) | PROJEKT (PR)
READ &ANS
IF &ANS = &STR() THEN GOTO B4
IF &ANS = &STR(PROJEKT) | &ANS = &STR(PR) THEN GOTO B2
IF &ANS = &STR(NAME) |  &ANS = &STR(NM) THEN GOTO B3
GOTO B1
B2: WRITENR PROJEKT / &USER / =:
READ &ANS
```

```
IF &ANS ¬= &STR() THEN SET &USER = &ANS
GOTO B1
B3: WRITENR NAME / &BUCH / =:
READ &ANS
IF &ANS ¬= &STR() THEN SET &BUCH = &ANS
GOTO B1
B4: WRITE UFO-BIBLIOTHEK / &UFUS..&UFBU..LOAD / =:
WRITENR GUELTIGE ANTWORTEN : ENTER(=JA) | NAME (NM) | PROJEKT (PR)
READ &ANS
IF &ANS = &STR() THEN GOTO B9
IF &ANS = &STR(PROJEKT) | &ANS = &STR(PR) THEN GOTO B5
IF &ANS = &STR(NAME) |  &ANS = &STR(NM) THEN GOTO B6
SET &UFA = &STR()
GOTO B9
B5: WRITENR PROJEKT / &UFUS / =:
READ &ANS
IF &ANS ¬= &STR() THEN SET &UFUS = &ANS
GOTO B4
B6: WRITENR NAME / &UFBU / =:
READ &ANS
IF &ANS ¬= &STR() THEN SET &UFBU = &ANS
GOTO B4
B9: SET &SYSDVAL = +
        &USER&B&BUCH&B&UFUS&B&UFBU&B&GER&B&XFAC&B&YFAC&STR( END)
SET &L = &LENGTH(&SYSDVAL)
OPENFILE OUTDA OUTPUT
SET &OUTDA=&SUBSTR(1:&L,&SYSDVAL)
PUTFILE OUTDA
CLOSFILE OUTDA
FREE F(OUTDA)
WRITE BEGINN DER PROZEDUR
CONCAT ('SYS7.FORTLIB' 'SYS2.FORTLIB' 'SYS2.GS7')
ALLOC DD(BB) DS('&USER..&BUCH') SHR
IF &UFA ¬= &STR() THEN +
  ALLOC DD(UFOLIB) DS('&UFUS..&UFBU..LOAD') SHR
M1: WRITENR NAME DES BILDES =: ( "ENTER"=STOP | "?" BILD-LISTE )
  READ BIN
IF &BIN¬=&STR(?) THEN GOTO M2
  LM '&USER..&BUCH'
  GOTO M1
M2: IF &BIN=&STR() THEN GOTO M7
M3: WRITENR GROESSE DES PLOTTERFENSTERS DIN-A =:
READ DIN
IF &DATATYPE(&DIN)¬=NUM THEN GOTO M3
IF &DIN > 10 THEN SET &DIN=10
M4: WRITENR FORMAT ( H | Q ) =:
READ FORM
IF &FORM=&STR() THEN GOTO M4
SET FORM=&SUBSTR(1,&FORM)
IF &FORM=H THEN +
   DO
   IF &DIN<2 THEN SET DIN=2
   SET X0=1923
   SET Y0=0
   SET X1=4095
   SET Y1=3071
```

```
   GOTO M5
   END
ELSE IF &FORM=Q THEN +
   DO
   IF &DIN<1 THEN SET DIN=1
   SET X0=0
   SET Y0=0
   SET X1=4095
   SET Y1=2896
   GOTO M5
   END
ELSE GOTO M4
/* BILDKONTROLLE AM GRAPHISCHEM ANSCHLUSS
M5: INITGS 1024
    RG GA
    RPIB &BIN BB
    ROUT &X0 &Y0 &X1 &Y1
    ENDGS
M6: WRITENR AUSGABE =: ( "ENTER"=WEG | OUT | ROUT )
READ AUS
IF &AUS=&STR() THEN GOTO M1
IF &AUS¬=OUT && &AUS¬=ROUT THEN GOTO M6
SET NB=&NB+1
SET &BIN&NB=&BIN
SET &DIN&NB=&DIN
SET &AUS&NB=&AUS
SET &FORM&NB=&FORM
IF &NB<5 THEN GOTO M1
WRITE MAXIMALZAHL ZU PLOTTENDER BILDER (5) ERREICHT
M7: IF &NB=0 THEN GOTO E3
   IF &GER ¬= &STR(VER)  THEN GOTO M10
/* PAPIER-VORGABE BEIM VERSATEC-PLOTTER
SET &XMAX=&XSTAN
M8: WRITE XMAX / &XMAX / =:
READ ANS
IF &ANS > &XSTAN THEN GOTO M9
IF &ANS = &STR() THEN GOTO M10
M9: SET &XMAX = &ANS
GOTO M8
M10: WRITENR JOB-KENNZEICHNUNG =:
READ JB
IF &LENGTH(&JB) > 2 THEN SET &JB = &SUBSTR(1:2,&JB)
IF &JB = &STR()     THEN SET &JB = &STR(X)
SET &JB=&BEN&JB
SET &L=&LENGTH(&JB)
/* EDITIEREN DES BACKGROUNDJOBS
  EDIT 'TSO105.SERVUS.CNTL(BRAUSJOB)' CNTL
  C 10 ?_NN_?&JB
  C 10 ?_NO_?&BEN
  C 110 ?_XF_?&XFAC
  C 110 ?_YF_?&YFAC
  C 200 220 ?_GG_?&GER? ALL
  C 210 ?_US_?&USER
  C 210 ?_D_?&BUCH
  C 230 340 ?_10_?&BIN1? ALL
  C 260 350 ?_11_?&DIN1? ALL
```

```
  C 250 350 ?_12_?&FORM1? ALL
  C 310 350 ?_13_?&AUS1? ALL
  C 380 ?_15_?&NB
  C 390 500 ?_20_?&BIN2? ALL
  C 420 510 ?_21_?&DIN2? ALL
  C 410 510 ?_22_?&FORM2? ALL
  C 470 510 ?_23_?&AUS2? ALL
  C 540 ?_25_?&NB
  C 550 660 ?_30_?&BIN3? ALL
  C 580 670 ?_31_?&DIN3? ALL
  C 570 670 ?_32_?&FORM3? ALL
  C 630 670 ?_33_?&AUS3? ALL
  C 700 ?_35_?&NB
  C 710 820 ?_40_?&BIN4? ALL
  C 740 830 ?_41_?&DIN4? ALL
  C 730 830 ?_42_?&FORM4? ALL
  C 790 830 ?_43_?&AUS4? ALL
  C 860 ?_45_?&NB
  C 870 980 ?_50_?&BIN5? ALL
  C 900 990 ?_51_?&DIN5? ALL
  C 890 990 ?_52_?&FORM5? ALL
  C 950 990 ?_53_?&AUS5? ALL
  IF &GER=&STR(VER)  THEN GOTO E1
  C 1170 ?PARM?TAPE
  SET &IDENT=&QU&SUBSTR(4:&L,&JB)
  C 1170 ?*?UNIT=T0800,LABEL=(,NL),DCB=DEN=2,VOL=SER=&IDENT
  DEL 1180
  GOTO E2
  E1: C 1180 ?_LL_?&XMAX
  E2: C 1190 ?_US_?&USER
  C 1190 ?_D_?&BUCH
  IF &UFA¬= &STR()  THEN  +
  1200 //UFOLIB DD DSN=&UFUS..&UFBU..LOAD,DISP=SHR
  IF &GER= &STR(VER)  THEN 1210 //B EXEC SVPLOT,SPACE=20
  SUBMIT
  END NOSAVE
E3: FREE DD(BB)
IF &UFA ¬= &STR()  THEN FREE DD(UFOLIB)
EXIT
```

Die Prozedur BRAUS erzeugt ein Stapel-Auftrag , indem sie anhand der erfragten Eingabe die CNTL-Datei BRAUSJOB abändert und zur Ausführung abschickt. Es folgt die Liste dieser Datei:

```
//_NN_ JOB (0abc,xyz,p9x9y),benutzer,MSGLEVEL=(1,1),NOTIFY=_NO_,          00000010
// MSGCLASS=H                                                             00000020
//A EXEC F7CLG,PLOT=GS7,SPACE=20                                          00000030
//C.SYSIN DD *                                                           00000040
C-----154B7, BUSCHMANN, AUG. 1978, IBM 370, FORTRAN 77 (1983 - 84)       00000050
C-----HAUPTPROGRAMM ZUM WIEDEREINLESEN UND PLOTTEN VON BIS ZU 5 EXTERN   00000060
C-----GESPEICHERTEN GS-BILDERN                                           00000070
      REAL G(10)/7.07107E-1,5.E-1,3.53553E-1,2.5E-1,1.76777E-1,1.25E-1   00000080
     =,8.83883E-2,6.25E-2,4.41942E-2,3.125E-2/                           00000090
      CHARACTER*8 INFO(8)                                                00000100
      DATA XFAC/_XF_/,YFAC/_YF_/,DIN0/46.84/,DIN1/33.12/                 00000110
      CALL JOBINF(INFO)                                                  00000120
```

```
      INFO(3)(1:1) = ' '                                                    00000130
      INFO(4)(4:8) = '     '                                                00000140
      INFO(6)(1:2) = '  '                                                   00000150
      WRITE(6,1000) INFO(3),INFO(4),INFO(6),INFO(7),INFO(8)                 00000160
 1000 FORMAT(/' DATUM DES JOBS:',2X,3A8,2X,2A8 )                            00000170
      CALL INITFG(IERR,1536)                                                00000180
      WRITE(6,1050) XFAC , YFAC                                             00000190
 1050 FORMAT(/' _GG_-PLOT',5X,'X-FAKTOR =',F6.2,5X,'Y-FAKTOR =',F6.2/       00000200
     ;/' DS-NAME DES BUCHES = _US_._D_'/)                                   00000210
      CALL RG('_GG__')                                                      00000220
  100 CALL RPIB('_10__','BUCH_')                                            00000230
      IF(IERR.NE.0)          GO TO 910                                      00000240
      IF('_12_'.EQ.'Q')         GO TO 170                                   00000250
      X=DIN1*G(_11_)*XFAC                                                   00000260
      Y=DIN0*G(_11_)*YFAC                                                   00000270
          GO TO 180                                                         00000280
  170 X=DIN0*G(_11_)*XFAC                                                   00000290
      Y=DIN1*G(_11_)*YFAC                                                   00000300
  180 CALL _13_(0.0,0.0,X,Y)                                                00000310
      IF(IERR.NE.0)          GO TO 920                                      00000320
      WRITE(6,199)  X,Y                                                     00000330
  199 FORMAT(' NAME DES BILDES    = _10_'/' FORMAT',13X                     00000340
     ;,'= DIN A _11_ _12_'/' AUSGABEFUNKTION    = _13_ '                    00000350
     ;,' ( 0." , 0." ,',F8.4,'" ,',F8.4,'" )' )                             00000360
      CALL DPI('* ')                                                        00000370
      IF(_15_.EQ.1)             GO TO 930                                   00000380
  200 CALL RPIB('_20__','BUCH_')                                            00000390
      IF(IERR.NE.0)          GO TO 910                                      00000400
      IF('_22_'.EQ.'Q')         GO TO 270                                   00000410
      X=DIN1*G(_21_)*XFAC                                                   00000420
      Y=DIN0*G(_21_)*YFAC                                                   00000430
          GO TO 280                                                         00000440
  270 X=DIN0*G(_21_)*XFAC                                                   00000450
      Y=DIN1*G(_21_)*YFAC                                                   00000460
  280 CALL _23_(0.0,0.0,X,Y)                                                00000470
      IF(IERR.NE.0)          GO TO 920                                      00000480
      WRITE(6,299) X,Y                                                      00000490
  299 FORMAT(' NAME DES BILDES    = _20_'/' FORMAT',13X                     00000500
     ;,'= DIN A _21_ _22_'/' AUSGABEFUNKTION    = _23_ '                    00000510
     ;,' ( 0." , 0." ,',F8.4,'" ,',F8.4,'" )' )                             00000520
      CALL DPI('* ')                                                        00000530
      IF(_25_.EQ.2)             GO TO 930                                   00000540
  300 CALL RPIB('_30__','BUCH_')                                            00000550
      IF(IERR.NE.0)          GO TO 910                                      00000560
      IF('_32_'.EQ.'Q')         GO TO 370                                   00000570
      X=DIN1*G(_31_)*XFAC                                                   00000580
      Y=DIN0*G(_31_)*YFAC                                                   00000590
          GO TO 380                                                         00000600
  370 X=DIN0*G(_31_)*XFAC                                                   00000610
      Y=DIN1*G(_31_)*YFAC                                                   00000620
  380 CALL _33_(0.0,0.0,X,Y)                                                00000630
      IF(IERR.NE.0)          GO TO 920                                      00000640
      WRITE(6,399) X,Y                                                      00000650
  399 FORMAT(' NAME DES BILDES    = _30_'/' FORMAT',13X                     00000660
     ;,'= DIN A _31_ _32_'/' AUSGABEFUNKTION    = _33_ '                    00000670
     ;,' ( 0." , 0." ,',F8.4,'" ,',F8.4,'" )' )                             00000680
```

```
      CALL DPI('* ')                                                    00000690
      IF(_35_.EQ.3)              GO TO 930                              00000700
  400 CALL RPIB('_40__','BUCH_')                                        00000710
      IF(IERR.NE.0)              GO TO 910                              00000720
      IF('_42_'.EQ.'Q')          GO TO 470                              00000730
      X=DIN1*G(_41_)*XFAC                                               00000740
      Y=DIN0*G(_41_)*YFAC                                               00000750
          GO TO 480                                                     00000760
  470 X=DIN0*G(_41_)*XFAC                                               00000770
      Y=DIN1*G(_41_)*YFAC                                               00000780
  480 CALL _43_(0.0,0.0,X,Y)                                            00000790
      IF(IERR.NE.0)              GO TO 920                              00000800
      WRITE(6,499) X,Y                                                  00000810
  499 FORMAT(' NAME DES BILDES    = _40_'/' FORMAT',13X                 00000820
     ;,'= DIN A _41_ _42_'/' AUSGABEFUNKTION    = _43_ '                00000830
     ;,' ( 0." ,0." ,',F8.4,'" ,',F8.4,'" )' )                          00000840
      CALL DPI('* ')                                                    00000850
      IF(_45_.EQ.4)              GO TO 930                              00000860
  500 CALL RPIB('_50__','BUCH_')                                        00000870
      IF(IERR.NE.0)              GO TO 910                              00000880
      IF('_52_'.EQ.'Q')          GO TO 570                              00000890
      X=DIN1*G(_51_)*XFAC                                               00000900
      Y=DIN0*G(_51_)*YFAC                                               00000910
          GO TO 580                                                     00000920
  570 X=DIN0*G(_51_)*XFAC                                               00000930
      Y=DIN1*G(_51_)*YFAC                                               00000940
  580 CALL _53_(0.0,0.0,X,Y)                                            00000950
      IF(IERR.NE.0)              GO TO 920                              00000960
      WRITE(6,599) X,Y                                                  00000970
  599 FORMAT(' NAME DES BILDES    = _50_'/' FORMAT',13X                 00000980
     ;,'= DIN A _51_ _52_'/' AUSGABEFUNKTION    = _53_ '                00000990
     ;,' ( 0." ,0." ,',F8.4,'" ,',F8.4,'" )' )                          00001000
      CALL DPI('* ')                                                    00001010
          GO TO 930                                                     00001020
  910 WRITE(6,9900)                                                     00001030
 9900 FORMAT(/' *** ABBRUCH DA BILD NICHT GEFUNDEN')                    00001040
          GO TO 930                                                     00001050
  920 WRITE(6,9920)IERR                                                 00001060
 9920 FORMAT(/' *** ABBRUCH DA GS-FEHLER ',Z8,' IN OUT/ROUT')           00001070
  930 CALL ENDGS(NB)                                                    00001080
      WRITE(6,9950)NB                                                   00001090
 9950 FORMAT(/' GROESSTER VON GS BEANSPRUCHTER WORKSPACE = ',I4,        00001100
     1' K BYTES')                                                       00001110
      WRITE(6,9999)                                                     00001120
 9999 FORMAT(/' ENDE DES JOBS')                                         00001130
      STOP                                                              00001140
      END                                                               00001150
//G.COMM DD DUMMY                                                       00001160
//G.PLOTPARM DD *                                                       00001170
 &PLOT XMAX=_LL_,MSGLVL=2, &END                                         00001180
//G.BUCH DD DSN=_US_._D_,DISP=SHR                                       00001190
//                                                                      00001220
```

11.8 Die Kommandoprozedur PIXI.

PIXI braucht ein bis zwei Buch-Dateien. Für die Prozedur-Eingaben
ist die Datei MEMORY(PIXI) vorgesehen .

Name : tso000.SERVUS.CLIST(PIXI)

```
PROC 0
CONTROL MAIN MSG PROMPT
WRITE PROZEDUR      PIXI
WRITE EINSTIEG ZUM ZEICHNEN MIT DEM GRAPHISCHEN SYSTEM ( GS ) .
/* EINGBEDATEN HOLEN UND UEBERPRUEFEN
SET &B = &STR( )
ALLOC FILE(OUTDA) DA('tso000.MEMORY(PIXI)')
OPENFILE OUTDA
GETFILE OUTDA
SET &SYSDVAL=&OUTDA
CLOSFILE OUTDA
READDVAL &USER1 &BUCH1 &USER2 &BUCH2 &UFUSER &UFBUCH
WRITE
WRITE ERSTE BILDER DATEI , "B1" :
WRITENR PROJEKT / &USER1 / =: ( "ENTER" = JA )
READ &US1
IF &US1 EQ &STR() THEN GOTO E1
SET &USER1 = &US1
E1: WRITENR NAME / &BUCH1 / =:
READ &BU1
IF &BU1 EQ &STR() THEN GOTO E2
SET &BUCH1 = &BU1
E2: WRITE ZWEITE BILDER DATEI , "B2" :
WRITENR PROJEKT / &USER2 / =: ( "ENTER"(=JA) | # (=KEIN "B2") )
READ &US2
IF &US2 EQ &STR(#) THEN GOTO E4
IF &US2 EQ &STR() THEN GOTO E3
SET &USER2 = &US2
E3: WRITENR NAME / &BUCH2 / =:
READ &BU2
IF &BU2 EQ &STR() THEN GOTO E4
SET &BUCH2 = &BU2
E4: WRITE UFO-BUCH :
WRITENR PROJEKT / &UFUSER / =: ( "ENTER"(=JA) | # (=KEIN UFOBUCH) )
READ &UFS
IF &UFS EQ &STR(#) THEN GOTO E10
IF &UFS EQ &STR() THEN GOTO E5
SET &UFUSER = &UFS
E5: WRITENR NAME / &UFBUCH / =:
READ &UFB
IF &UFB EQ &STR() THEN GOTO E10
SET &UFBUCH=&UFB
E10: SET &SYSDVAL = +
```

```
  &USER1&B&BUCH1&B&USER2&B&BUCH2&B&UFUSER&B&UFBUCH&STR( END)
SET &L = &LENGTH(&SYSDVAL)
OPENFILE OUTDA OUTPUT
SET &OUTDA=&SUBSTR(1:&L,&SYSDVAL)
PUTFILE OUTDA
CLOSFILE OUTDA
FREE F(OUTDA)
WRITE BEGINN DER PROZEDUR
WRITE                                                  ****
WRITE DIE PROZEDUR KANN MAN DURCH  EINTIPPEN VON   STOP   BEENDEN.
WRITE                                                  ****
CONCAT ('SYS7.FORTLIB' 'SYS2.FORTLIB' 'SYS2.GS7')
ALLOC F(COMM)  DS(*)
ALLOC DD(GAAPLIB) DS('TSOSYS.GAS.FONT') SHR
ALLOC DA('&USER1..&BUCH1.') F(B1)
IF &US2 EQ &STR(#) THEN GOTO P1
ALLOC DA('&USER2..&BUCH2.') F(B2)
P1: IF &UFS EQ &STR(#) THEN GOTO P2
ALLOC DD(UFOLIB)  DS('&UFUSER..&UFBUCH..LOAD') SHR
P2: INITGS 3000
RG GAT
TERMIN STOP
ENDGS
FREE DD(GAAPLIB)
FREE DA('&USER1..&BUCH1.')
IF &UFS ¬= &STR(#) THEN    FREE DD(UFOLIB)
IF &US2 = &STR(#) THEN GOTO P8
FREE DA('&USER2..&BUCH2.')
WRITENR COMPRESS &BUCH2 ? =: ( "ENTER" = JA )
READ &ANS
IF &ANS ¬= &STR() THEN GOTO P8
COMPRESS '&USER2..&BUCH2.' NOLIST
REDUCE '&USER2..&BUCH2.'
SPACE '&USER2..&BUCH2.'
P8: WRITENR COMPRESS &BUCH1 ? =: ( "ENTER" = JA )
READ &ANS
IF &ANS ¬= &STR() THEN GOTO P9
COMPRESS '&USER1..&BUCH1.' NOLIST
REDUCE '&USER1..&BUCH1.'
SPACE '&USER1..&BUCH1.'
P9: EXIT
```

11.9 D i e K o m m a n d o p r o z e d u r L O A D .

Für die Eingabedaten wird die Datei MEMORY(LOAD) benötigt.

Name : tso000.SERVUS.CLIST(LOAD)

```
PROC 0
CONTROL MAIN MSG PROMPT
WRITE
WRITE PROZEDUR      LOAD
WRITE ERSTELLUNG EINES LOAD - MODULS ZU EINER SERVUS-DIALOG-PROZEDUR
WRITE
/* EINGBEDATEN HOLEN UND UEBERPRUEFEN
SET &B = &STR( )
ALLOC FILE(OUTDA) DA('tso000.MEMORY(LOAD)')
OPENFILE OUTDA
GETFILE OUTDA
SET &SYSDVAL=&OUTDA
CLOSFILE OUTDA
READDVAL &PROF &NAMF &GLIED &PROL &NAML
I1: WRITENR DIE QUELL-("FORT")-DATEI / &GLIED / =:  ( "ENTER"=JA )
     READ &ANS
IF &ANS=&STR() THEN GOTO J1
SET &GLIED = &ANS
J1: SET &SYSDVAL = &PROF&B&NAMF&B&GLIED&B&PROL&B&NAML&STR( END)
SET &L = &LENGTH(&SYSDVAL)
OPENFILE OUTDA OUTPUT
SET &OUTDA=&SUBSTR(1:&L,&SYSDVAL)
PUTFILE OUTDA
CLOSFILE OUTDA
FREE F(OUTDA)
WRITE BEGINN DER PROZEDUR
WRITE BENOETIGTE HILFSMODULE VOM SYSTEM ? =:
WRITENR ( "ENTER"=KEINE | TRACEGS (TR) | +
 ALLES ANDERE = NUR SERVUS )
     READ &ANS
IF &ANS = &STR() THEN GOTO K5
IF &ANS = &STR(TRACEGS) | &ANS = &STR(TR) THEN GOTO K3
CONCAT ('INR105.SERVUS.LOAD' 'SYS2.FORTLIB' 'SYS7.FORTLIB')
WRITE DEM MODUL WERDEN DIE SERVUS-MODULE ZUGEFUEGT
GOTO K5
K3: CONCAT ('INR105.SERVUS.LOAD' 'SYS2.FORTLIB' 'SYS7.FORTLIB' +
            'SYS2.TRACEGS7' 'SYS2.GS7')
WRITE DEM MODUL WERDEN DIE TRACEGS- UND DIE SERVUS-MODULE ZUGEFUEGT
K5: SET &NAME = &STR(SERVUS.FORT(&GLIED.))
SET &PROG = &GLIED
COPY &NAME &PROG..FORT
FORT77 &PROG. NOGO OBJECT(&PROG..OBJ) NOPRINT LANGLVL(77)
LINK &PROG..OBJ  LET  PRINT(*) LOAD('INR105.SERVUS.LOAD(&GLIED.)')
WRITE MODULNAME : 'INR105.SERVUS.LOAD(&GLIED.)'
ERASE &PROG..FORT
ERASE &PROG..OBJ
WRITE ENDE DER PROZEDUR LOAD
EXIT
```

11.10 D i e K o m m a n d o p r o z e d u r T E X T .

Name : tso000.SERVUS.CLIST(TEXT)

```
PROC 0 ALF(1) OM(80)
WRITE PROZEDUR
WRITE                DAS SERVUS-HANDBUCH
WRITE       WIRD VON DER SEITE &ALF BIS ZUR SEITE &OM
WRITE            AM SCHNELLDRUCKER AUSGEGEBEN .
WRITE
GLOBAL NAME1 NAME2 NAME3 NAME4
CONTROL MAIN PROMPT MSG
WRITENR VON SEITE / &ALF / =:
READ &ANS
IF &ANS EQ &STR() THEN GOTO E1
SET &ALF = &ANS
E1: WRITENR BIS SEITE / &OM / =:
READ &O
IF &O EQ &STR() THEN GOTO E2
SET &OM = &O
E2: ATTRIB TEXT BLKSIZE(3120) LRECL(255) DSORG(PS) RECFM(V B)
ALLOC DA('&SYSPREF..Q.TEXT') NEW SPACE(5,10) TRA RELEASE USING(TEXT)
FREE ATTRLIST(TEXT)
CONTROL MAIN PROMPT MSG END(STOP) LIST
ALLOC FILE(INTEX) DA('INR105.SERVUS.TEXT(SERVUS)')  OUTPUT
OPENFILE INTEX
GETFILE INTEX
SET &I = 1
       DO  WHILE &I < 15
       ERROR GOTO E3
       MERGE 'INR105.&INTEX' 1  999  '&SYSPREF..Q.TEXT' NONUM NONUM2
       GETFILE INTEX
       SET &I = &I+1
       STOP
E3: CLOSFILE INTEX
FREE DS('INR105.SERVUS.TEXT')
EDIT '&SYSPREF..Q.TEXT' OLD TEXT NONUM ASIS
RENUM
C 10 25000 !% ! ! ALL
C 10 25000 !¢ (! (! ALL
C 10 25000 !¢ )! )! ALL
UNNUM
END SAVE
TEXTP '&SYSPREF..Q.TEXT' DRUCK VON(&ALF) BIS(&OM) FMT(A) STZ(@) UML( ¢ )
ERASE '&SYSPREF..Q.TEXT'
EXIT
```

Literatur:

/ 1/ Fachausdrücke der Text- und Datenverarbeitung
IBM Deutschland 1978.

/ 2/ K. Gogg , J. Buschmann u. a. , GS-Handbuch.
Unveröffentlichter Bericht des KfK , Dez. 1982.

/ 3/ C. Broeders , PLOTDAT - Programmbeschreibung ,
Unveröffentlichter Bericht 1978.

/ 4/ M. Politzky, TRACEGS , Ein Programmsystem für die Zeichenungs-
erstellung..., KFK 3237 ,1981.

/ 5/ H. Kleinheins , DDPLOT - Programmbeschreibung ,
Unveröffentlichter Bericht 1978.

/ 6/ SIEMENS AKTIENGESELLSCHAFT , Operating System BS3000
FORTRAN77 User's Guide
Version 1 , September 1982.
Chapt. 15. List-directed READ statement. P. 295

Anhang S. Aufbau der Datei INR105.SERVUS.LOAD

1. Member : IDAJOB
Bestimmung : Eine SERVUS-Datei wird neu angelegt oder umbenannt .
Enthält die Routinen : IDAJOB
NESZE

2. Member : ENDE
Bestimmung : Schreibt eine Endzeile in eine Datei .
Enthält die Routine : ENDE

3. Member : SERDIO
Bestimmung : Schreibt Funktionen in eine Datei ("SDIN") bzw.
liest Funktionen einer Datei ("SDEX") .
Enthält die Routinen : SERDIO
SDINPY
SDING4
SDING8
SDINF8
SDEXPY
SDEXG4
SDEXG8
SDEXF8

4. Member : DATAS
Bestimmung : Eine Datei wird im Dialogbetrieb gelesen/kopiert .
Enthält die Routinen : DATAS
NO

5. Member : COPYDA
Bestimmung : Eine Datei wird kopiert oder ausgedruckt ,
Funktionen einer Datei werden gelöscht ,
alles im Stapelbetrieb .
Enthält die Routinen : COPYDA
PAGEFF
ERASE

6. Member : NEXT
Bestimmung : Sucht aus die nächste Funktion,
Verteilt die Ausgabe auf mehreren Seiten .
Enthält die Routinen : NEXT
CIFFRA
PAGINA
ORDNE

7. Member : FENST4
Bestimmung : Korrigiert die Funktionen beim Kopieren .
Enthält die Routinen : FENST4
FENST8
NUMDY4
NUMDY8
VALS4
VALS8
CIRCA

8. Member : FIGUR
Bestimmung : Funktionen einer Datei werden als Kurven gezeigt
(Dialogbetrieb) .
Enthält die Routine : FIGUR

9. Member : FIGURB
Bestimmung : Funktionen einer Datei werden als Kurven gezeigt
(Stapelbetrieb) .
Enthält die Routine : FIGURB

10. Member : PAPIER
Bestimmung : Papieraufteilung , Netzherstellung , Achsengestaltung
bei der Bildherstellung.
Enthält die Routinen : PAPIER
FENCO
AXON

11. Member : CORTEX
Bestimmung : Beginn und Ende eines Bildes , sowie Textkorrektur
im Dialogverfahren.
Enthält die Routine : CORTEX
PINIT
FINIP
GSNAME

12. Member : DDFIG
Bestimmung : Funktionen einer Datei werden als Fläche gezeigt
(Dialogbetrieb) .
Enthält die Routine : DDFIG

13. Member : DDFIGB
Bestimmung : Funktionen einer Datei werden als Fläche gezeigt
(Stapelbetrieb) .
Enthält die Routine : DDFIGB

14. Member : WUERFL
Bestimmung : Hilfsroutinen für die Flächen-Darstellungen .
Enthält die Routinen : WUERFL
MIRROR
AREA
DDPLO1
SICHT
STUECK
DDAXIS
AXTEX
FAULA

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                               +
+      8.11.84     LIST OF THE GRAPHS OF THE DATASET       VIELBILD.DATA        +
+                                                                               +
+                         STORAGE-NORM IS      "GRAPHIC4"                       +
+                                                                               +
+     DATE OF THE ALLOCATION:    3. 5.84    LAST MODIFICATION   30.10.84       +
+                                                                               +
+                       THE DATASET CONTAINS   310 GRAPHS                       +
+                                                                               +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

ORDNUNGSZAHLEN DER KURVEN :   ALLE  IM  BEREICH    1 -   99

 #      1   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : + P PI'  + *1.E+ 6+
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  100.0      , U= 10.
F1=  89.64    ,F(100)=  89.64     ,   89.00     < F <  150.0      , U= 10.

 #      2   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DM(W<S) + *1.E- 9+
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  100.0      , U= 10.
F1= 0.0       ,F(100)= 0.0        ,  -1.600     < F < 0.8200      , U=0.10

 #      3   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(RO,G)+    K   +
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  100.0      , U= 10.
F1= 0.3168E-07,F(100)= 0.3378E-07 , -0.1900     < F < 0.1100      , U=0.10E-01

 #      4   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(S<F) + *1.E+ 3+
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  100.0      , U= 10.
F1=  2.470    ,F(100)=  2.470     ,   1.300     < F <  7.600      , U= 1.0

 #      5   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(G<S) +   ERG  +
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  100.0      , U= 10.
F1= 0.1485E-01,F(100)= 0.1424E-01 , -20.00      < F <  22.00      , U= 1.0
...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  .........

...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  .........
 #     98   GRAPH(   3/   95)  X : + HOEHE  +   CM   +  F : +DY(RY,G)+ *1.E- 6+
X1= 0.0       ,X(100)=  99.00     ,  0.0        < X <  99.00      , U= 10.
F1= 0.0       ,F(100)= 0.0        ,  -3.500     < F <  3.100      , U=0.50

 #     99   GRAPH(   3/   95)  X : + HOEHE  +   CM   +  F : +E (G)   + *1.E- 3+
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  100.0      , U= 10.
F1=  44.53    ,F(100)=  48.87     ,  0.0        < F <  100.0      , U= 10.

              LIST OF THE FUNCTIONS FROM THE     1. TO THE    99.
```

Abb. 1. Katalog-Auszug der Datei 'INR105.VIELBILD.DATA'

```
PROGRAMM   P A G E F F

IM BEREICH     1  <  J  < 1000     WIRD JEDE     1 -TE FUNKTIONSWERT AUSGEDRUCKT.

DIE ERSTE FUNKTION ERSCHEINT MIT ORDINATEN UND ABSZISSEN ,
                        BEI DEN ANDEREN WERDEN NUR DIE ORDINATEN AUSGEDRUCKT.


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+     13.11.84     LIST OF THE GRAPHS OF THE DATASET       SCDAR3   .DATA      +
+                                                                              +
+                         STORAGE-NORM IS       "GRAPHIC8"                     +
+                                                                              +
+     DATE OF THE ALLOCATION:    6. 5.83    LAST MODIFICATION   22. 7.83      +
+                                                                              +
+                        THE DATASET CONTAINS   216 GRAPHS                     +
+                                                                              +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

ORDNUNGSZAHLEN DER KURVEN :      4   -5   -3   -6  -13  -14   -1

> > > > > > >  BEGINN DES BLATTES NR   1    GESUCHT DIE KURVE    4  > > > > > >

 #      4   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(S<F) +    ERG  +
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  99.00     , U= 10.
F1=  2470.    ,F(100)=  2470.     ,   2267.     < F <  2470.     , U= 41.

- - - - - - -  FORTS. DES BLATTES NR   1    GESUCHT DIE KURVE    5  - - - - - -

 #      5   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(G<S) +    ERG  +
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  99.00     , U= 10.
F1= 0.1485D-01,F(100)= 0.1424D-01 ,  -19.97     < F <  4.813     , U= 5.0

- - - - - - -  FORTS. DES BLATTES NR   1    GESUCHT DIE KURVE    3  - - - - - -

 #      3   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(RO,G)+     K   +
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  99.00     , U= 10.
 ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
 ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
 ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...

F1= 0.1485D-01,F(100)= 0.1424D-01 , -0.2912D-01 < F < 0.1584D-01 , U=0.90D-02

- - - - - - -  FORTS. DES BLATTES NR   1    GESUCHT DIE KURVE    1  - - - - - -

 #      1   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : + P PIN  + ERG/CCM+
X1= 0.5000    ,X(100)=  99.50     ,  0.0        < X <  99.00     , U= 10.
F1= 0.8964D+08,F(100)= 0.8964D+08 ,  0.8964D+08 < F < 0.1453D+09 , U=0.11D+08
```

Abb. 2. Ausdruck-Spiegel , erste Seite .

B L A T T # 1

	HOEHE CM (3/ 85)	DW(S<F) ERG (3/ 85)	DW(G<S) ERG (3/ 85)	DW(RO,G) K (3/ 85)	DW(G<S) K (3/ 85)	-DV1*WA K (3/ 85)	OMEGA K (3/ 85)	P PIN ERG/CCM (3/ 85)
1	0.50000000	2470.2293	0.14848984D-01	0.31683518D-07	0.21579710D-01	-0.57410204D-02	0.14848984D-01	89640492.
2	1.5000000	2470.2293	0.14848984D-01	0.31683518D-07	0.21579710D-01	-0.57410204D-02	0.14848984D-01	89640492.
3	2.5000000	2470.2293	4.8128792	0.31683518D-07	0.21579710D-01	-0.57410204D-02	0.15838690D-01	89640492.
4	3.5000000	2147.3559	4.3036059	-0.54984986D-06	0.19853007D-01	-0.20258648D-01	-0.40564087D-03	91954232.
5	4.5000000	1854.6601	4.5328658	-0.16669238D-05	0.21610241D-01	-0.40578645D-01	-0.18968404D-01	94204098.
6	5.5000000	1589.6315	4.4654673	-0.40117483D-05	0.22129569D-01	-0.60933873D-01	-0.38804303D-01	96402027.
7	6.5000000	1350.3041	5.7439081	-0.62139314D-05	0.29673326D-01	-0.71904621D-01	-0.42231295D-01	98558729.
8	7.5000000	2043.5987	6.7680780	-0.13458868D-05	0.35060226D-01	-0.61340373D-01	-0.26280146D-01	0.10057001D+09
9	8.5000000	2779.2999	6.7587089	0.33121170D-05	0.35050893D-01	-0.32574244D-01	0.24766493D-02	0.10253158D+09
10	9.5000000	3493.9357	6.8503968	0.79047019D-05	0.35561728D-01	-0.11989703D-02	0.34362758D-01	0.10444820D+09
11	10.500000	4190.9663	6.9179210	0.98057399D-05	0.35948496D-01	0.16477233D-01	0.52425729D-01	0.10632178D+09
12	11.500000	4873.1089	6.9711096	0.75266890D-05	0.36260993D-01	0.12879560D-01	0.49140553D-01	0.10815381D+09
13	12.500000	5542.3860	7.0052583	0.23118515D-05	0.36474559D-01	0.53200604D-02	0.31154498D-01	0.10994543D+09
14	13.500000	6200.1444	7.0097696	-0.12799624D-05	0.36534153D-01	-0.20583536D-01	0.15950617D-01	0.11169744D+09
15	14.500000	6847.0413	7.0797057	0.29339222D-05	0.36930900D-01	-0.15888835D-01	0.21042065D-01	0.11341037D+09
16	15.500000	7482.9440	6.5205651	0.17753111D-04	0.34071231D-01	0.15508295D-01	0.49579527D-01	0.11508453D+09
17	16.500000	7570.7308	6.0141751	0.55939715D-04	0.32269865D-01	0.68139167D-01	0.10040903	0.11672057D+09
18	17.500000	7366.9993	5.8312644	0.16437041D-03	0.32283764D-01	0.12774557	0.16002933	0.11831811D+09
19	18.500000	7162.7354	5.8411555	0.37180484D-03	0.32870414D-01	0.17924918	0.21211960	0.11987659D+09
20	19.500000	6957.5529	5.8526207	0.69430374D-03	0.32933825D-01	0.21615927	0.24909309	0.12139563D+09
21	20.500000	6751.1679	5.8840481	0.11694298D-02	0.33229390D-01	0.23970335	0.27293274	0.12287475D+09
22	21.500000	6543.4228	5.9293474	0.18421319D-02	0.33930624D-01	0.24831491	0.28224553	0.12431339D+09
23	22.500000	6334.2844	5.9892389	0.26949143D-02	0.34827128D-01	0.23542454	0.27025167	0.12571091D+09
24	23.500000	6123.8404	6.0470604	0.35813287D-02	0.35691880D-01	0.19499700	0.23068888	0.12706663D+09
25	24.500000	5912.2952	6.0580528	0.41722459D-02	0.36283785D-01	0.12669160	0.16297538	0.12837981D+09
26	25.500000	5699.9645	5.9473338	0.39295844D-02	0.36174470D-01	0.36877114D-01	0.73051584D-01	0.12964971D+09
27	26.500000	5487.2690	5.6207804	0.21451725D-02	0.34712272D-01	-0.63876491D-01	-0.29164219D-01	0.13087554D+09
28	27.500000	5274.7274	4.9799225	-0.19307024D-02	0.31202[illegible]D-01	-0.16464607	-0.13344358	0.13205651D+09
29	28.500000	5062.9480	3.9344770	-0.89972561D-02	0.24995881D-01	-0.25655554	-0.23155966	0.13319185D+09
30	29.500000	4852.6197	2.4232265	-0.19568574D-01	0.15599744D-01	-0.33155080	-0.31595106	0.13428076D+09
31	30.500000	4644.5021	0.46043001	-0.33645496D-01	0.30016405D-02	-0.38039788	-0.37739624	0.13532246D+09
32	31.500000	4439.4151	-1.7893182	-0.50181331D-01	-0.11805412D-01	-0.39326044	-0.40506585	0.13631619D+09
33	32.500000	4238.2278	-3.9519189	-0.66628690D-01	-0.26371147D-01	-0.36367501	-0.39004616	0.13726120D+09
34	33.500000	4041.8473	-5.4663028	-0.79035282D-01	-0.36870812D-01	-0.29324983	-0.33012064	0.13815674D+09
35	34.500000	3851.2071	-5.7184558	-0.82933400D-01	-0.38968272D-01	-0.19339641	-0.23236468	0.13900212D+09
36	35.500000	3667.2551	-4.2440778	-0.74760181D-01	-0.29207467D-01	-0.82582806D-01	-0.11179027	0.13979663D+09
37	36.500000	3490.9423	-0.90252481	-0.53115790D-01	-0.62685403D-02	0.19067742D-01	0.12799201D-01	0.14053961D+09
38	37.500000	3323.2115	4.0227674	-0.19645166D-01	0.28166699D-01	0.94676054D-01	0.12284275	0.14123042D+09
39	38.500000	3164.9853	9.8105798	0.20690075D-01	0.69179412D-01	0.13372147	0.20290088	0.14186845D+09
40	39.500000	3017.1561	15.402861	0.60318063D-01	0.10928314	0.13337635	0.24265949	0.14245312D+09
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
50	49.500000	2266.8343	-19.970591	-0.18506996	-0.13388402	0.10476699	-0.29117029D-01	0.14526636D+09

Abb. 3. Ausdruck-Spiegel , zweite Seite .

B L A T T # 1

	HOEHE CM (3/ 85)	DW(S<F) ERG (3/ 85)	DW(G<S) ERG (3/ 85)	DW(RO,G) K (3/ 85)	DW(G<S) K (3/ 85)	-DV1*WA K (3/ 85)	OMEGA K (3/ 85)	P PIN ERG/CCM (3/ 85)
51	50.500000	2274.7503	-18.732101	-0.17759594	-0.12628310	0.93059906D-01	-0.33223195D-01	0.14523798D+09
52	51.500000	2298.4194	-14.962237	-0.15389016	-0.10231165	0.60220827D-01	-0.42090819D-01	0.14515287D+09
53	52.500000	2337.6928	-8.7761105	-0.11280954	-0.61059227D-01	0.14468386D-01	-0.46590840D-01	0.14501112D+09
54	53.500000	2392.3243	-0.86625042	-0.57787651D-01	-0.61202636D-02	-0.29338986D-01	-0.35459250D-01	0.14481289D+09
55	54.500000	2461.9727	7.4921365	0.21795160D-02	0.53499110D-01	-0.53977706D-01	-0.47859573D-03	0.14455839D+09
56	55.500000	2546.2059	14.763124	0.55166575D-01	0.10597454	-0.47765155D-01	0.58209381D-01	0.14424789D+09
57	56.500000	2644.5050	19.592716	0.90475943D-01	0.14073791	-0.10772640D-01	0.12996527	0.14388171D+09
58	57.500000	2756.2701	21.199852	0.10212478	0.15189639	0.44680654D-01	0.19657704	0.14346024D+09
59	58.500000	2880.8264	19.573254	0.90266287D-01	0.13962142	0.99156080D-01	0.23877750	0.14298388D+09
60	59.500000	3017.4319	15.401450	0.60309970D-01	0.10927274	0.13336736	0.24264010	0.14245312D+09
61	60.500000	3165.2852	9.8069573	0.20666486D-01	0.69153582D-01	0.13370440	0.20285798	0.14186845D+09
62	61.500000	3323.5347	4.0179821	-0.19676586D-01	0.28133061D-01	0.94655349D-01	0.12278841	0.14123042D+09
63	62.500000	3491.2879	-0.90741870	-0.53147444D-01	-0.63024975D-02	0.19039619D-01	0.12737121D-01	0.14053961D+09
64	63.500000	3667.6220	-4.2480513	-0.74784806D-01	-0.29234637D-01	-0.82618798D-01	-0.11185344	0.13979663D+09
65	64.500000	3851.5943	-5.7207067	-0.82945562D-01	-0.38983370D-01	-0.19343061	-0.23241398	0.13900212D+09
66	65.500000	4042.2539	-5.4665596	-0.79033295D-01	-0.36872305D-01	-0.29326076	-0.33013306	0.13815674D+09
67	66.500000	4238.6525	-3.9505901	-0.66615534D-01	-0.26362099D-01	-0.36364124	-0.39000334	0.13726120D+09
68	67.500000	4439.8569	-1.7872748	-0.50163010D-01	-0.11791844D-01	-0.39317998	-0.40497183	0.13631619D+09
69	68.500000	4644.9599	0.46234186	-0.33627686D-01	0.30140810D-02	-0.38029230	-0.37727822	0.13532246D+09
70	69.500000	4853.0924	2.4245028	-0.19554558D-01	0.15607832D-01	-0.33145311	-0.31584528	0.13428076D+09
71	70.500000	5063.4346	3.9349753	-0.89879258D-02	0.24998831D-01	-0.25649102	-0.23149219	0.13319185D+09
72	71.500000	5275.2267	4.9797022	-0.19257121D-02	0.31200829D-01	-0.16462265	-0.13342182	0.13205651D+09
73	72.500000	5487.7801	5.6199617	0.21464686D-02	0.34706888D-01	-0.63892666D-01	-0.29185778D-01	0.13087554D+09
74	73.500000	5700.4864	5.9460901	0.39281274D-02	0.36166541D-01	0.36819815D-01	0.72986356D-01	0.12964971D+09
75	74.500000	5912.8271	6.0566074	0.41693343D-02	0.36274782D-01	0.12658920	0.16286398	0.12837981D+09
76	75.500000	6124.3813	6.0456318	0.35782929D-02	0.35683141D-01	0.19485915	0.23054229	0.12706663D+09
77	76.500000	6334.8335	5.9879644	0.26926501D-02	0.34819370D-01	0.23528826	0.27010763	0.12571091D+09
78	77.500000	6543.9794	5.9282361	0.18409070D-02	0.33923938D-01	0.24823693	0.28216087	0.12431339D+09
79	78.500000	6751.7314	5.8830834	0.11689788D-02	0.33223749D-01	0.23972323	0.27294698	0.12287475D+09
80	79.500000	6958.1228	5.8513334	0.69427745D-03	0.32926722D-01	0.21628227	0.24920899	0.12139563D+09
81	80.500000	7163.3112	5.8419372	0.37174474D-03	0.32874633D-01	0.17933801	0.21221264	0.11987659D+09
82	81.500000	7367.5808	5.8183458	0.16491131D-03	0.32212038D-01	0.12828220	0.16049423	0.11831811D+09
83	82.500000	7571.2418	5.9704783	0.57014447D-04	0.32033391D-01	0.69732846D-01	0.10176624	0.11672057D+09
84	83.500000	7483.0159	6.4831660	0.18523746D-04	0.33874226D-01	0.17030227D-01	0.50904454D-01	0.11508453D+09
85	84.500000	6847.0413	7.0683237	0.31397500D-05	0.36872059D-01	-0.15528614D-01	0.21343445D-01	0.11341037D+09
86	85.500000	6200.1444	7.0118110	-0.13117953D-05	0.36544[illegible]3D-01	-0.20727269D-01	0.15817435D-01	0.11169744D+09
87	86.500000	5542.3860	7.0056214	0.22726086D-05	0.36476443[illegible]-01	-0.54163862D-02	0.31060057D-01	0.10994543D+09
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
100	99.500000	2470.2293	0.14237720D-01	0.33781635D-07	0.21656574D-01	-0.59739371D-02	0.14237720D-01	89640492.

< < < < < < < < < < < < < < < ENDE DES BLATTES NR 1 < < < < < < < < < < < < < < <

Abb. 4. Ausdruck-Spiegel , dritte Seite .

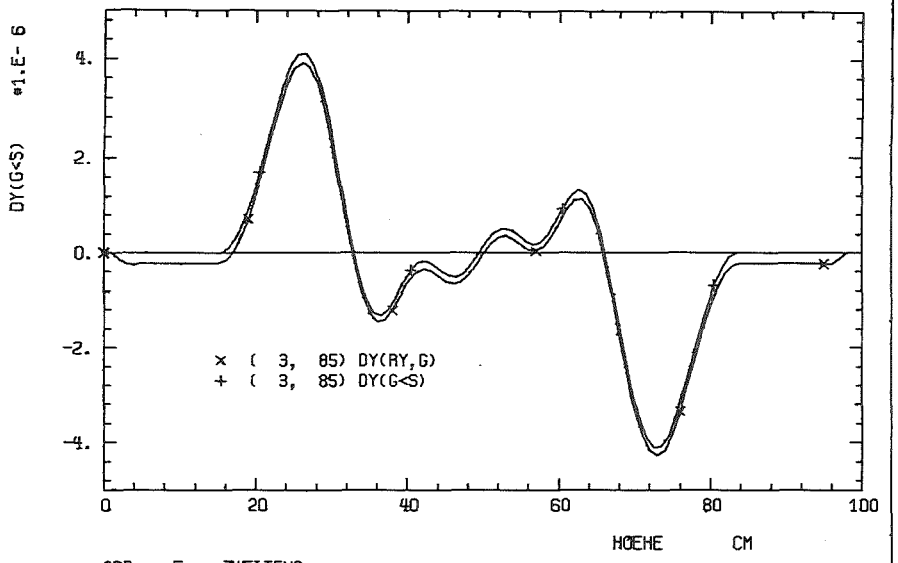


ABB. 5. ZWEITENS

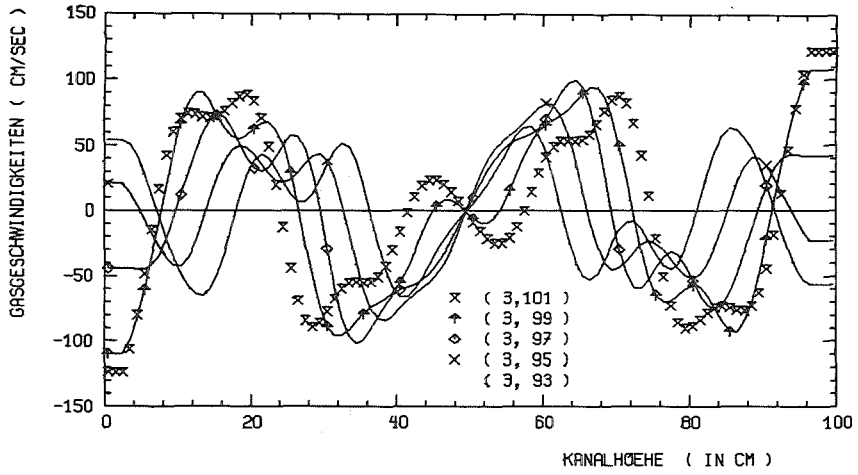


Abb. 5. Entwicklung der Gasgeschwindigkeiten im Kühlkanal.

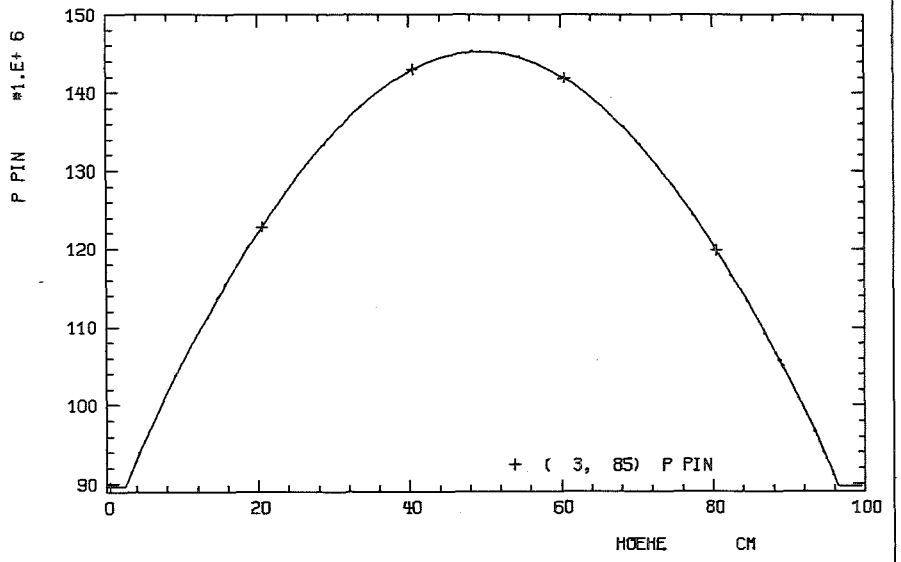


ABB. 1.

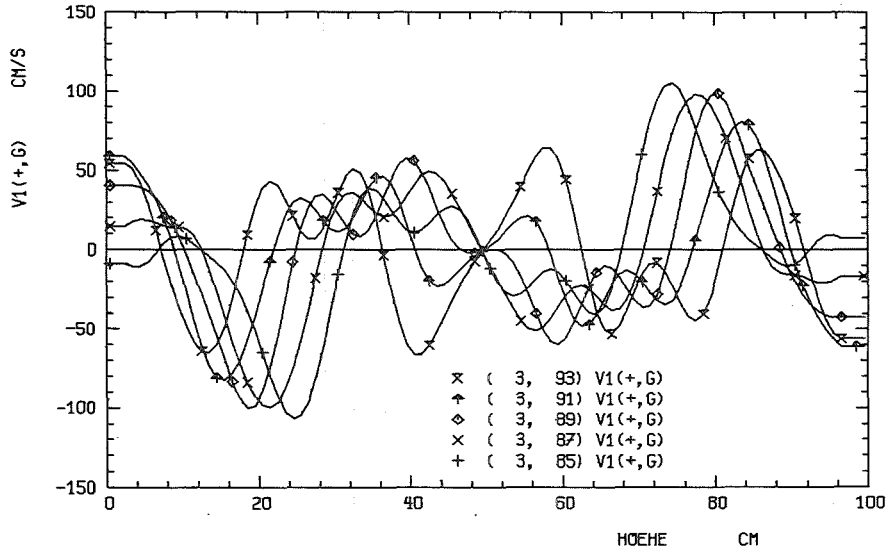
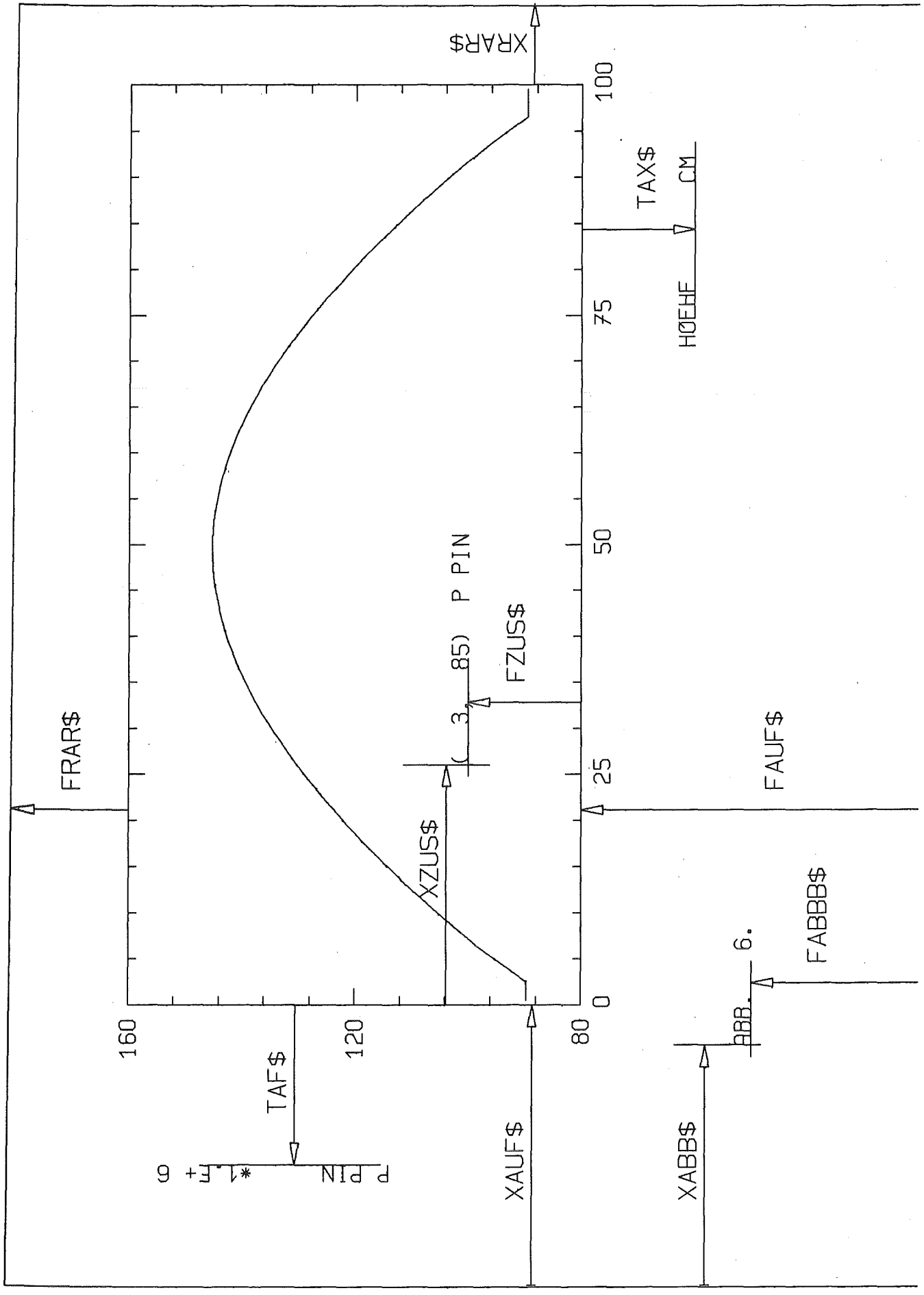


ABB. 5. SECHSTENS


XRAR$
100
75
50
25
0
80
120
160
TAX$
CM
HØEHF
FZUS$
85) P PIN
3.
XZUS$
FRAR$
FAUF$
FABBB$
6.
BBB.
XABB$
XAUF$
TAF$
P PIN
*1.E+ 6

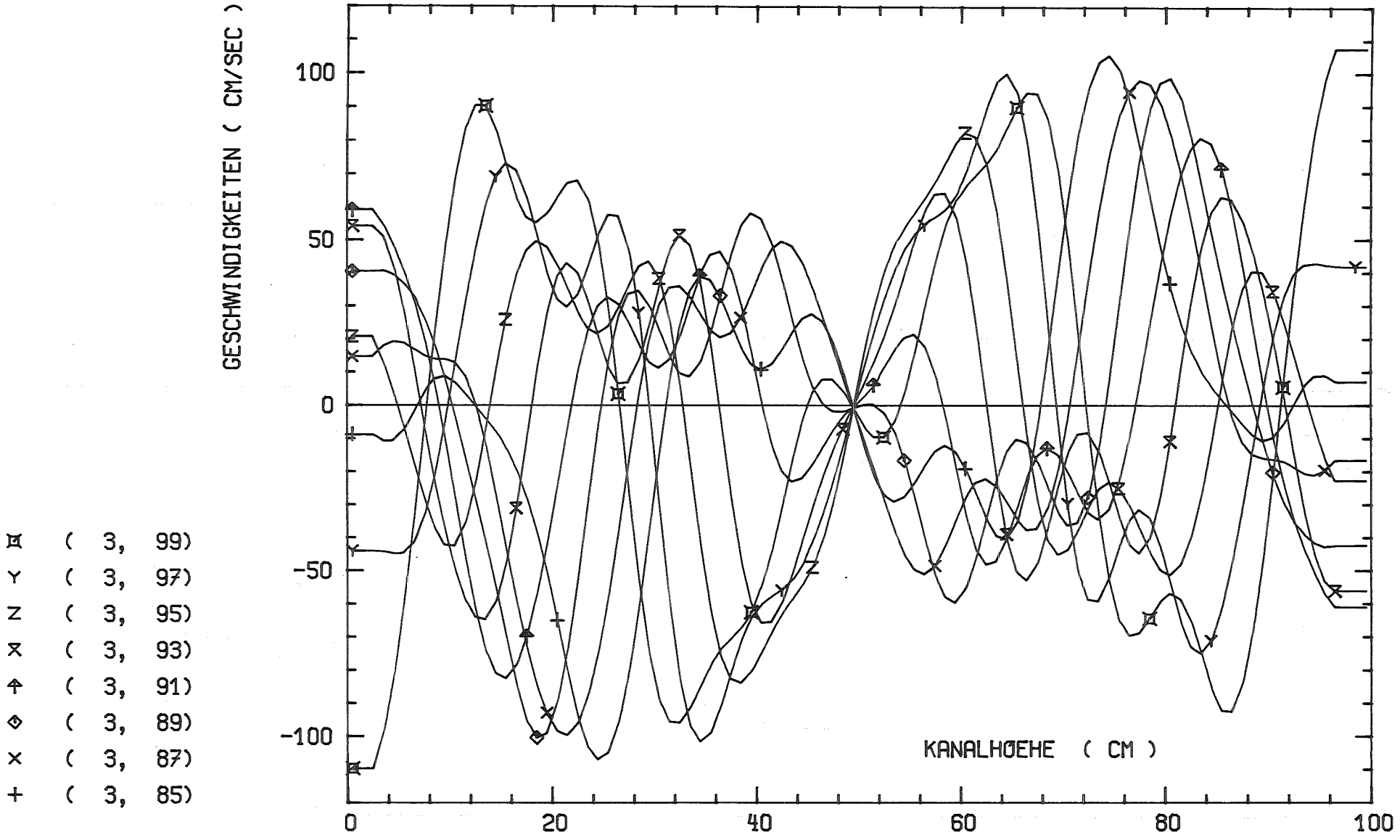


Abb. 7. Entwicklung der Gasgeschwindigkeiten im Kühlkanal.

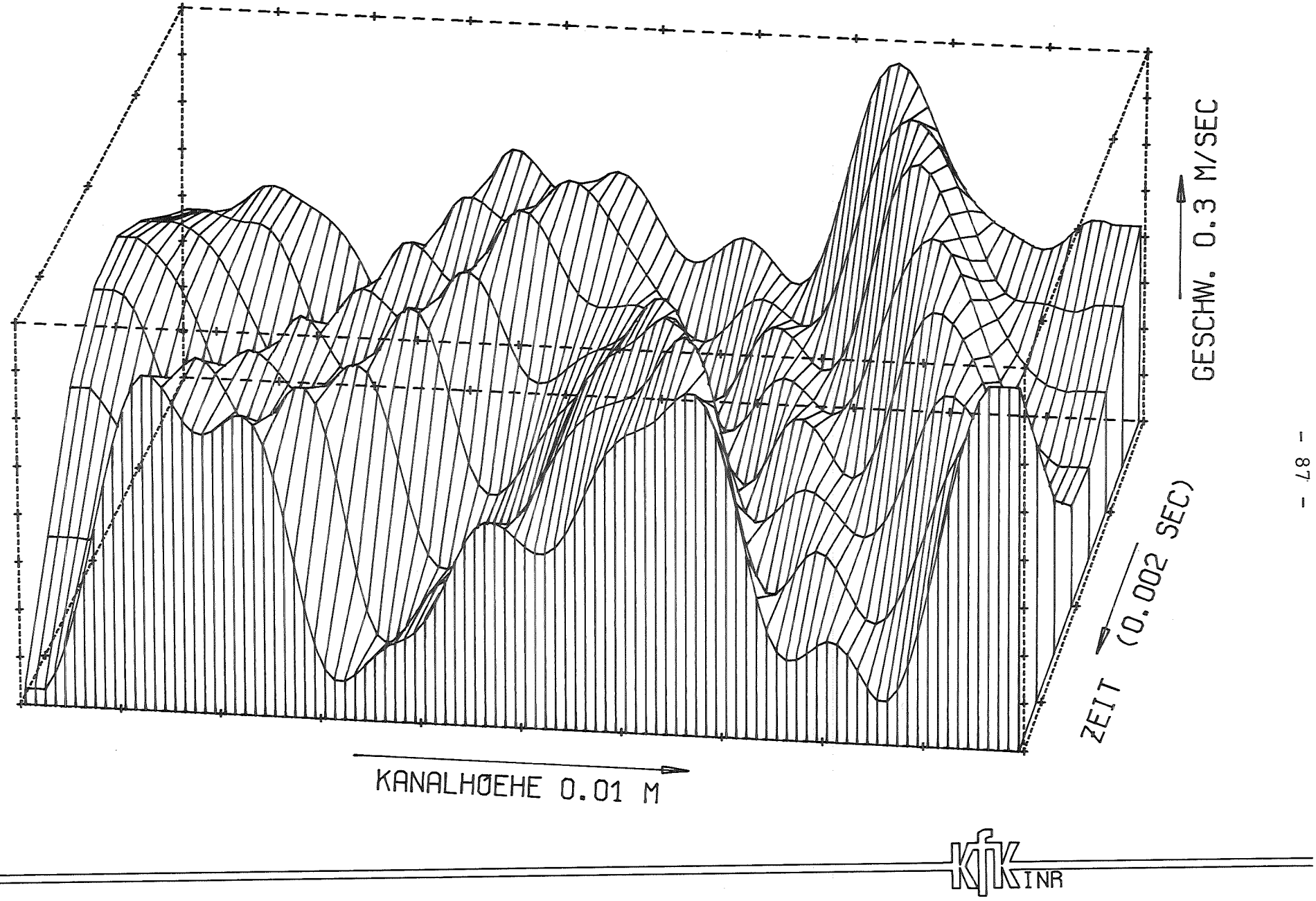


Abb. 8. Entwicklung der Gasgeschwindigkeiten im Kühlkanal.

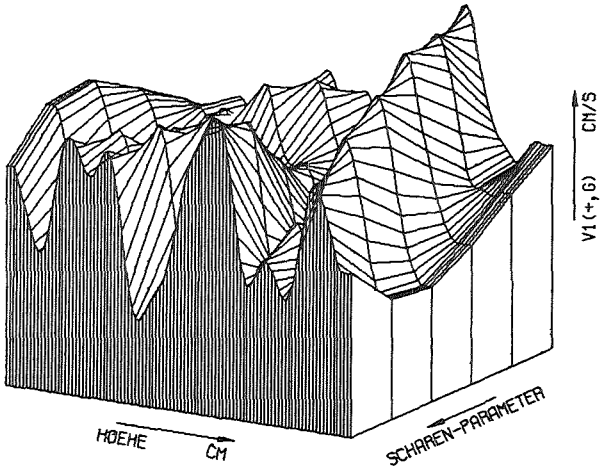


ABB. 9. ZWEITENS

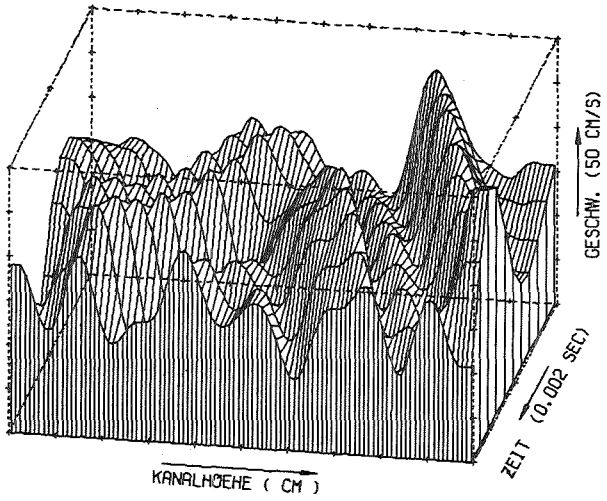


Abb. 9. Entwicklung der Gasgeschwindigkeiten im Kühlkanal.

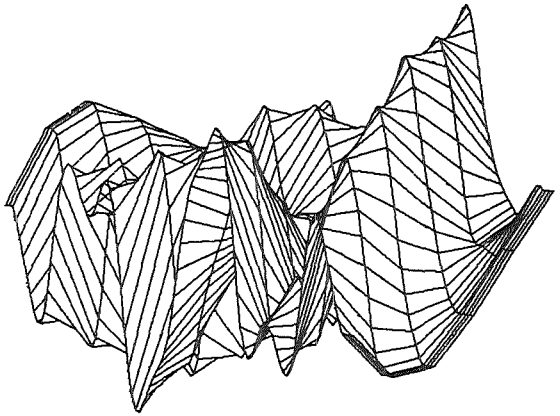


ABB. 1.

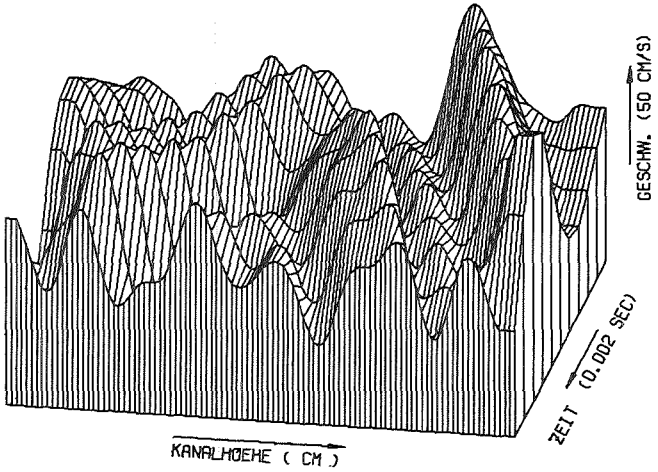


ABB. 9. SECHSTENS

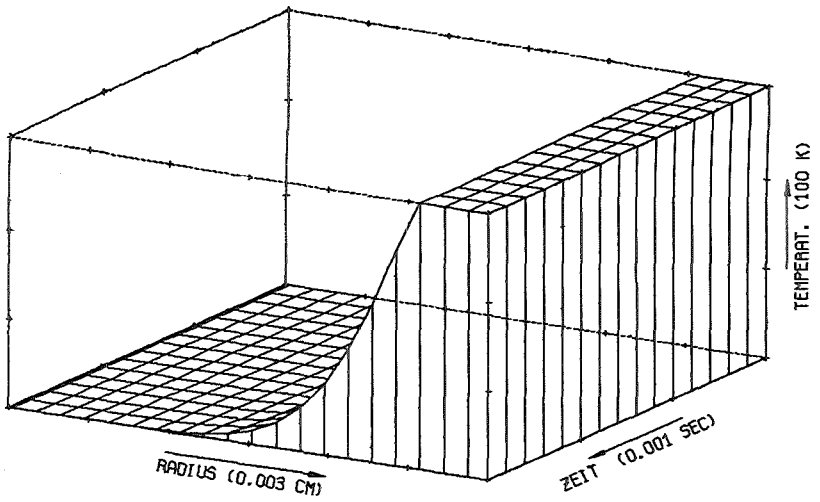


ABB. 10. DIE KOMPLEMENTAERE OBERFLAECHE.

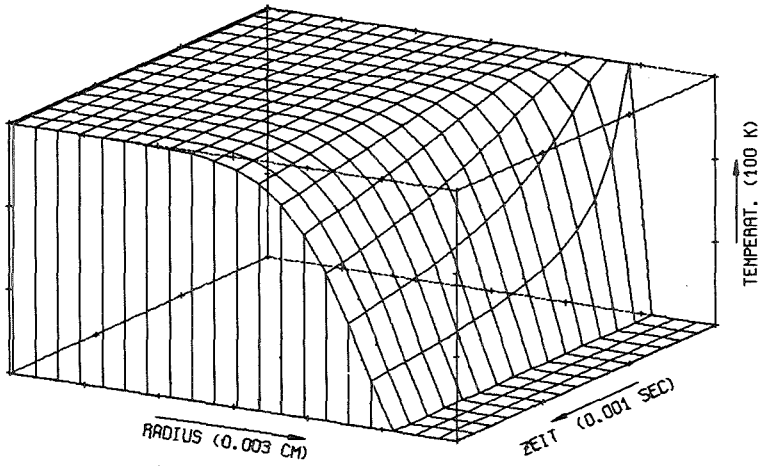


ABB. 10. ABKUEHLUNG EINES STAHLZYLINDERS.

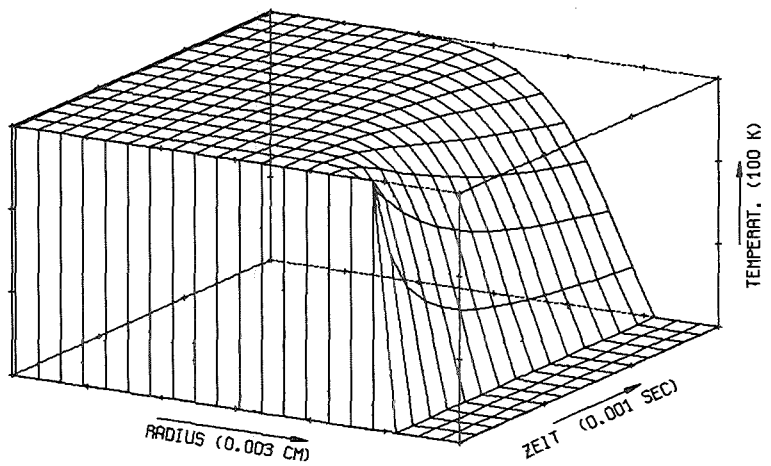


ABB. 10. DIE GESPIEGELTE OBERFLAECHE.