Programmierungsbeispiele und -Techniken für Ihren HP-42S Rechner

Dieses Handbuch enthält Beispiele aus den Bereichen Mathematik, Naturwissenschaft, Technik und Finanzmathematik. Es soll Ihnen dabei behilflich sein, maximalen Nutzen aus den eingebauten Applikationen Ihres Rechners zu gewinnen. *Programmierte* Lösungen werden dabei hervorgehoben, wobei auch Grafiken und Ausdrucke über den Infrarot-Taschendrucker HP 82240A angesprochen werden.

- **Programmierung**
  Einfache Programmierung • Verzweigungen • Schleifensteuerung • Indirekte Adressierung • Flags in Programmen • Fehlerabfragen
- **Erweiterung von HP-41 Programmen**
  Verwenden von benannten Variablen • Anwendung von HP-42S Datenein- und Ausgabefunktionen • Operationen mit HP-42S Datentypen • Verwenden der 2-zeiligen Anzeige • Verwenden von Menüvariablen • Programmzuweisung für CUSTOM Menüs
- **Der Löser**
  Allgemeine Anwendung des Lösers • Vorgabe von Anfangsnäherungen für den Löser • Emulation des Lösers • Anwendung des Lösers in Programmen • Einzelheiten zur Funktionsweise des Lösers
- **Integration**
  Normale Integration • Approximation eines Integrals mit Unendlichkeitsgrenze • Interaktive Anwendung von SOLVER und Integration • Einzelheiten zur Funktionsweise des Integrationsalgorithmus
- **Matrizen**
  Verwenden des Matrix-Editors und der Indizierungsfunktionen • Vektorrechnung • Lösen linearer Gleichungssysteme • Verwenden des Lösers für lineare Gleichungssysteme • Matrixoperationen in Programmen
- **Statistik**
  Statistische Berechnungen mit Listen • Verwenden der Summationsfunktionen in Programmen • Kurvenanpassung über Programme
- **Grafische Darstellung**
  Grafiken • Plotten mehrerer Funktionen • Ausdrucken von Daten einer komplexen Matrix

HEWLETT PACKARD

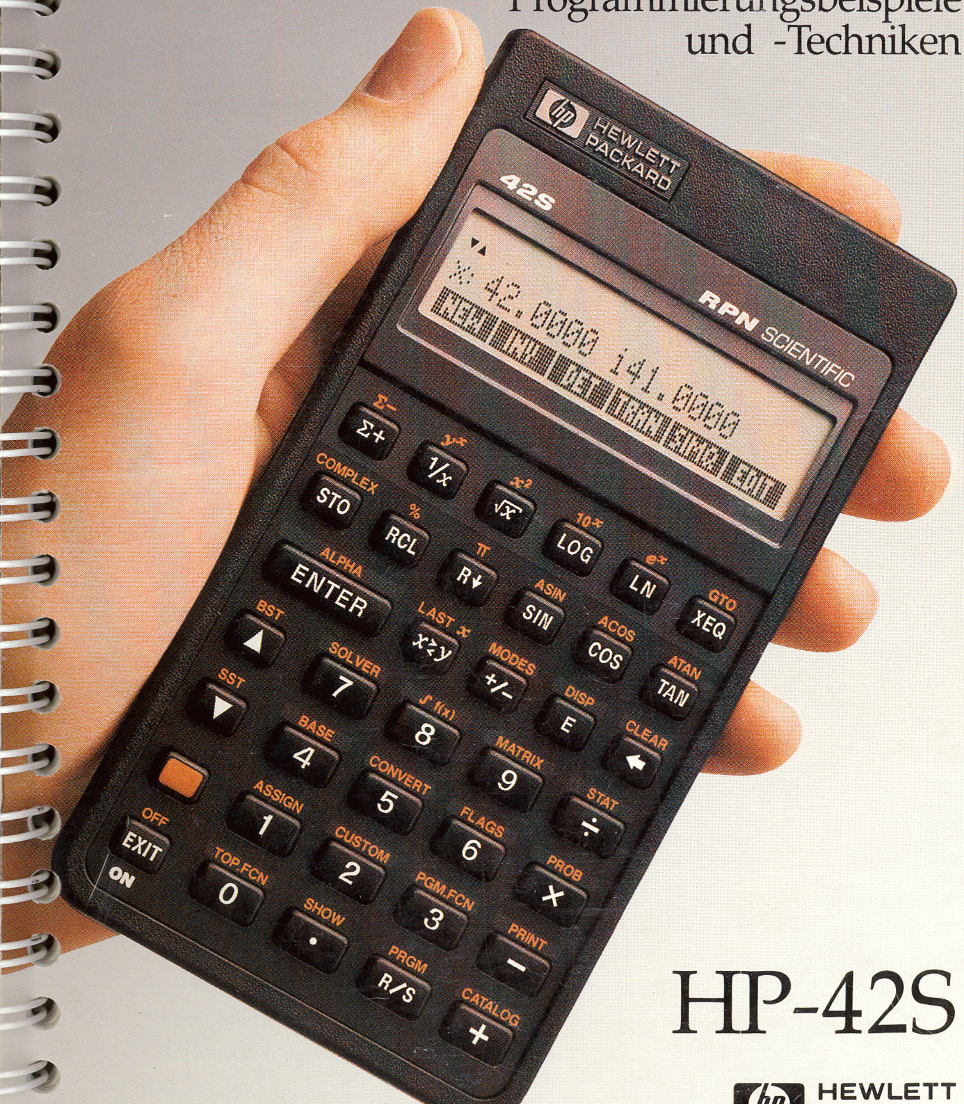
Bestellnummer
00042-90026

00042-90027 German
Printed in West Germany 10/88

HEWLETT-PACKARD

RPN Scientific Calculator

Programmierungsbeispiele und -Techniken



HP-42S

HEWLETT PACKARD

Eine kleine Anstrengung ...

Bitte nehmen Sie sich die Zeit, um diese Karte auszufüllen. Sie helfen damit Hewlett-Packard, Ihre Anforderungen besser zu verstehen. Lesen Sie zuerst alle Fragen durch, bevor Sie mit dem Ausfüllen beginnen. Vielen Dank!

Eine kleine Anstrengung ...

Handbuch: **Programmierungsbeispiele und -Techniken** Kaufdatum: ______

Name ____________________

Straße ____________________

PLZ/Ort ____________________

Tel-Nr. (________) ________________ Büro ☐ oder Priv. ☐

1. Für welchen Rechner verwenden Sie dieses Handbuch?
   009 ☐ **HP-42S** 006 ☐ **Anderes** ________________
2. Wieviel andere HP Lösungsbücher haben Sie für den HP-42S gekauft? ____
3. Was sind Sie von Beruf?
   101 ☐ **Student** 103 ☐ **Spezialist** 109 ☐ **Anderes** ________________
4. Wo haben Sie dieses Buch gekauft?
   403 ☐ **Buchhandlung** 404 ☐ **Kaufhaus**
   407 ☐ **Versandhandel** 410 ☐ **Direkt von HP** 411 ☐ **Anderes** __________
5. Wie haben Sie zum ersten Mal von diesem Handbuch gehört?
   501 ☐ **HP Besitzer** 503 ☐ **Anzeige** 506 ☐ **HP Verkäufer**
   507 ☐ **Prospekt/Broschüre** 508 ☐ **Anderes** ________________
6. Zu welchem Ausmaß hat dieses Handbuch zum Kauf des Taschenrechners beigetragen?
   601 ☐ **Großer Einfluß** 602 ☐ **Kleiner Einfluß** 603 ☐ **Kein Einfluß**
7. Wie gut deckt sich der Inhalt dieses Buchs mit Ihren Erwartungen?
   701 ☐ **Gut** 702 ☐ **Mittelmäßig** 703 ☐ **Kaum**
8. Welcher Kenntnisstand ist für die behandelten Themen erforderlich?
   801 ☐ **Hoch** 802 ☐ **Mittel** 803 ☐ **Nieder**
9. Wie klar wurden die Themen in diesem Handbuch dargestellt?
   901 ☐ **Gut** 902 ☐ **Mittelmäßig** 903 ☐ **Mäßig**
10. Wie schätzen Sie den Wert des Handbuches im Vergleich zum Kaufpreis?
    111 ☐ **Hoch** 112 ☐ **Mittel** 113 ☐ **Nieder**

Kommentar: (Bitte beziehen Sie sich auf Verbesserungen und zusätzliche Anwendungen oder Themen, welche HP in diesem oder einem anderen (Lösungs-) Handbuch behandeln sollte.) ________________

____

____

____

bitte freimachen

Postkarte

Antwort

Hewlett-Packard GmbH
Calculator Marketing
Hewlett-Packard-Str.
D-6380 Bad Homburg v.d.H.

HP-42S RPN Scientific

Programmierungsbeispiele und -Techniken

HEWLETT PACKARD

1. Ausgabe Oktober 1988
Bestellnummer 00042-90026

Hinweis

Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

Druckgeschichte

1. Ausgabe Oktober 1988 Fertigungsnr. 00042-90027

Inhaltsverzeichnis

Liste mit Beispielen

Die nachstehende Auflistung gruppiert die Beispiele kapitelweise.

Verwenden dieses Handbuchs

Das vorliegende Handbuch, *HP 42S Programmierungsbeispiele und -Techniken*, baut auf den im *HP-42S Benutzerhandbuch* eingeführten Konzepten auf und soll Ihnen dabei behilflich sein, von den leistungsstarken Fähigkeiten Ihres Rechners maximalen Gebrauch zu machen. Es wird dabei besonders auf die nachstehenden Sachgebiete eingegangen:

- Programmierungstechniken für den HP-42S.
- Erweiterung von vorhandenen HP-41 Programmen.
- Anwendung der internen HP-42S Applikationen:
  - SOLVER
  - Integration
  - Matrizen
  - Statistik
- Erzeugen und Drucken von Grafik-Mustern und Abbildungen.

Das vorliegende Handbuch enthält viele Beispiele, um Ihnen bei der Vertiefung Ihrer Kenntnisse über den Rechner größtmögliche Unterstützung zu bieten. Sie erkennen dabei auch, wie Lösungen für praktische Problemstellungen in den Gebieten Mathematik, Naturwissenschaft, Technik und Finanzen aufgefunden werden können. Viele Lösungen werden dabei unter Anwendung von Programmen erzielt. Kapitel 1, "Programmierung", beschäftigt sich mit dem Erstellen von Programmen im HP-42S. Außerdem werden Sachgebiete, die in den Kapiteln 8 bis 10 des Benutzerhandbuchs angesprochen sind, noch tiefgehender behandelt.

Kapitel 2 bezieht sich speziell auf die Erweiterung von Programmen, welche für den HP-41 geschrieben wurden. Es baut auf den Informationen auf, welche in Kapitel 11 des Benutzerhandbuchs vorgestellt werden.

Die Kapitel 3 bis 6 erläutern detailliert die internen Applikationen, welche in den Kapiteln 12 bis 15 des Benutzerhandbuchs besprochen sind. Wenn Sie mehr über Matrizenoperationen erfahren möchten, so können Sie sich auch direkt auf Kapitel 5, "Matrizen", konzentrieren, ohne die vorangehenden Kapitel zuerst durchzuarbeiten. Da jedoch viele der vorgestellen Lösungen in Programmen bestehen, sollten Sie auf jeden Fall Kapitel 1 durchlesen.

Kapitel 7 beschreibt das Erzeugen von Grafik-Mustern und von Abbildungen mit dem HP-42S, wobei in einigen Beispielen auch auf den optionalen Infrarot-Taschendrucker HP 82240A Bezug genommen wird. Die in Kapitel 7 des Benutzerhandbuchs enthaltenen Informationen werden dabei als Basis zugrundegelegt.

Die in diesem Handbuch verwendeten Notationen sind mit den im Benutzerhandbuch verwendeten Notationen konsistent:

- Für die in Tastenfolgen verwendeten Zahlen und Alphazeichen wird eine einfache Schriftart gewählt: 1.2345, ABCD.
- Für auf dem Tastenfeld enthaltene Erstfunktionen werden eingerahmte Tasten- bzw. Funktionsbezeichnungen verwendet: [EXIT].
- In oranger Schrift aufgedruckte Zweitfunktionen, auf welche durch die orange Umschalttaste zugegriffen wird, werden durch ein führendes schwarzes Quadrat (symbolisch für Umschalttaste) dargestellt: ■[ASSIGN].
- Menüfelder bzw. in Menüs enthaltene Funktionen sind durch ein dunkel hinterlegtes Rechteck mit entsprechendem Funktionsnamen dargestellt: CLP.
- Großgeschriebene Buchstaben dienen zur Kennzeichnung von Funktionsnamen, welche im erläuternden Text enthalten sind: CLP.
- Großgeschriebene Buchstaben dienen zur Kennzeichnung von Programmnamen, welche im erläuterndem Text enthalten sind: SSS.
- Kursive Schreibweise wird für Variablennamen verwendet, auf welche im Text Bezug genommen wird: *VAR1*
- Matrix-Schriftart wird zur Darstellung von Programmlisten verwendet:
  `01 LBL "KREIS"`.

Zu Beginn jedes Beispiels wird davon ausgegangen, daß die Stackregister (X-, Y-, Z- und T-Register) gelöscht sind (Inhalt = 0). Weiterhin wird vorausgesetzt, daß am Anfang der Inhalt jeder Variablen Null ist.

Der Anzeigeinhalt Ihres Rechners kann manchmal von dem im Handbuch dargestellten Inhalt abweichen. Wenn Sie jedoch die in den Beispielen enthaltenen Tastenfolgen ausführen, hat der Inhalt der Stackregister und der Variablen zu Beginn des Beispiels keine Auswirkung auf die erzielten Ergebnisse.

Einige Beispiele beinhalten optionale Anweisungen zum Drucken von Ergebnissen auf dem Infrarot-Taschendrucker HP 82240A. Wenn Sie über einen Drucker verfügen und diese Anweisungen ausführen, werden einige der nachfolgenden Anzeigen des Beispiels nicht angezeigt – sie werden *stattdessen gedruckt.*

1

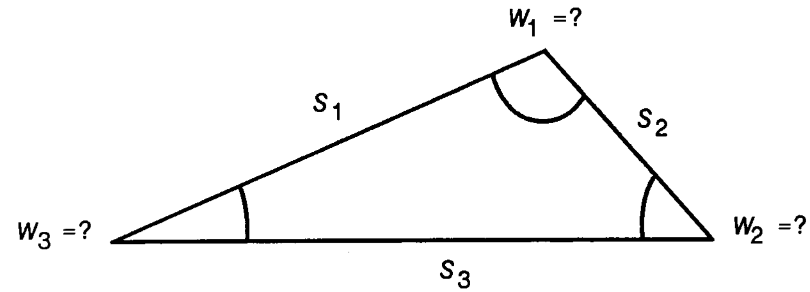
Programmierung

Ihr Rechner ist ein leistungsstarkes und sehr einfach zu handhabendes Werkzeug zum Erzeugen und Ausführen von Programmen. Dieses Kapitel baut auf den in den Kapiteln 8 bis 10 des Benutzerhandbuchs enthaltenen Informationen auf. Insbesondere wird hier auf nachstehende Themen eingegangen:

- Einfache Programmierung.
- Verzweigungen.
- Schleifensteuerung über Schleifenzähler.
- Indirekte Adressierung.
- Flags in Programmen.
- Fehlerabfrage und -Behandlung.

Einfache Programmierung

Das Programm SSS in diesem Abschnitt berechnet die Werte für die drei Winkel eines Dreiecks, wenn die Werte der drei Seiten vorgegeben sind. (Die zugehörige Programmliste finden Sie auf Seite 17 und 18.)



Sind die Dimensionen der drei Seiten (S_1, S_2 und S_3) eines Dreiecks bekannt, so können über nachstehende Gleichungen die zugehörigen drei Winkel (W_1, W_2, und W_3) berechnet werden.

$$W_3 = 2 \text{ arc cos} \left[\frac{\sqrt{P(P - S_2)}}{(S_1 S_3)} \right] \text{ wobei } P = \frac{(S_1 + S_2 + S_3)}{2}$$

$$W_2 = 2 \text{ arc cos} \left[\frac{\sqrt{P(P - S_1)}}{(S_2 S_3)} \right]$$

$$W_1 = \text{arc cos} [-\cos (W_3 + W_2)]^*$$

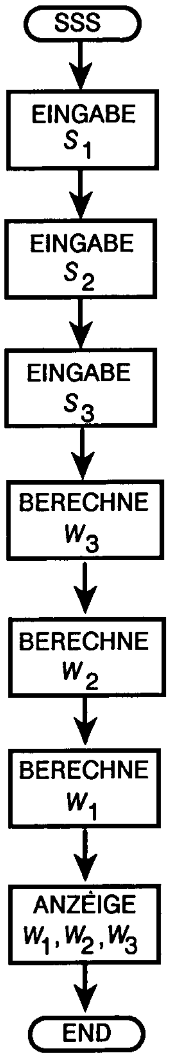
Diese Gleichungen bilden den Hauptteil von SSS.

Flußdiagramm

Die grafische Darstellung eines Algorithmus oder einer Problemstellung wird als Flußdiagramm (oder Ablaufdiagramm) bezeichnet. Flußdiagramme sollen Ihnen die Funktionsweise eines Programms veranschaulichen sowie beim Entwurf eigener Programme Unterstützung bieten. Ein Flußdiagramm kann so einfach oder so detailliert sein, wie Sie es wünschen. Es wird von oben nach unten gezeichnet, was dem Programmablauf von Anfang bis Ende entspricht.

* Dieser Ausdruck für W_1 erlaubt Ihnen die Berechnung von W_1 in einem beliebigen Winkelmodus.

Nachstehend ein Beispiel für eine mögliche Programmlösung des Seite-Seite-Seite Dreieckproblems.



Dieses Handbuch verwendet folgende Konventionen für Flußdiagramm-Symbole:

- Ein Oval stellt den *Beginn* oder das *Ende* von einer Routine dar. Hierbei kann es sich um ein Programm, eine Subroutine oder eine durch einen Zähler gesteuerte Schleife handeln.
- Ein Kreis stellt ein *Programm-Label* oder eine *GTO Anweisung* für ein Programm-Label einer anderen Stelle im Programm dar. (Diese Konvention reduziert die Notwendigkeit für Verbindungslinien, welche das Flußdiagramm schwerer lesbar machen können.)
- Ein Rechteck stellt eine *ausführende Operation* im Programm dar.
- Eine Raute stellt eine *Entscheidung* dar, die vom Programm aufgrund eines Vergleichs oder eines Flagstatus angestellt wird.
- Ein Dreieck stellt eine vom *Anwender* (d.h. Ihnen) getroffene Entscheidung dar, bei welcher eine Wahl zwischen mehreren möglichen Programmroutinen vorgenommen wird.

Definieren des Programms

Das Programm SSS beginnt mit einem *globalen Label* und endet mit einer END Anweisung. Diese beiden Anweisungen definieren den Beginn und das Ende des Programms.

```
01 LBL "SSS"
    :
45 END
```

Aufforderung zur Dateneingabe

SSS *fordert Sie zur Eingabe* von Zahlenwerten auf (die Werte der bekannten drei Seiten des Dreiecks).

```
02 INPUT "S1"
03 INPUT "S2"
04 INPUT "S3"
```

Anzeigen von Programmergebnissen

SSS schließt mit der *Anzeige* (oder dem Drucken) der berechneten Ergebnisse – drei Winkel – ab.

```
41 SF 21
42 VIEW "W1"
43 VIEW "W2"
44 VIEW "W3"
```

Dieser Abschnitt des Programms beginnt mit dem Setzen von Flag 21, Aktivieren des Druckers. Ist Flag 21 gesetzt, so wird eine VIEW (oder AVIEW) Anweisung:

- *gedruckt und angezeigt*, wenn PRON ausgeführt wurde. Das Programm stoppt nicht, nachdem eine Meldung angezeigt wurde; eine nachfolgende VIEW (oder AVIEW) Anweisung *überschreibt* die momentane Anzeige. Ist Flag 21 gesetzt und es wird PRON ausgeführt, so muß bei einer anschließenden Ausführung eines Programms, welches eine Folge von VIEW (oder AVIEW) Anweisungen enthält, ein Drucker vorhanden und eingeschaltet sein, um jede Meldung auszudrucken. Nur *die letzte* Meldung ist in der Anzeige ersichtlich.
- *angezeigt*, wenn PROFF ausgeführt wurde. (PROFF ist die Voreinstellung für den Rechner. Sie müssen PROFF nur dann ausführen, wenn zuvor PRON ausgeführt wurde.) Wird Flag 21 während des PROFF Modus gesetzt, hält das Programm nach jeder VIEW (oder AVIEW) Anweisung an und muß durch Drücken von [R/S] fortgesetzt werden.

Hilfreiche Hinweise zum Eintippen von Programmen:

1. Wenn die im Programm verwendeten Variablen nicht bereits existieren, sollten Sie diese *vor dem Umschalten in den Programmeingabe-Modus* erzeugen (durch Drücken von 0 [STO] *Variable* für jede Variable). Beim späteren Eintippen einer STO, RCL, INPUT, oder VIEW Anweisung während der Programmeingabe werden die existierenden Variablen angezeigt, wenn zur Angabe eines Registers aufgefordert wird. Sie müssen daraufhin lediglich die korrespondierende Menütaste drücken, anstatt den Variablennamen *einzutippen*.
2. Im Programmeingabe-Modus sind zuerst alle globalen Labels (durch Drücken von ■[PGM.FCN] LBL *Label*) für jedes Label einzugeben. Wenn Sie später eine Anweisung zum Verzweigen eintippen und zur Angabe eines Labels aufgefordert werden, erhalten Sie die existierenden globalen Labels im Programmkatalog-Menü angezeigt. Sie müssen daraufhin lediglich die korrespondierende Menütaste drücken, anstatt den Namen *einzutippen*.

In diesem Handbuch sind längeren Programmen die Anweisungen vorangestellt, welche die Variablen und Labels auflisten, die zur Programmeingabe erforderlich sind.

Um das Programm SSS einzutippen: Erzeugen Sie die Variablen *S1*, *S2*, *S3*, *W1*, *W2*, *W3* und *P*, bevor Sie mit der Programmeingabe beginnen.

Nachstehend die Programmliste von SSS.

Programm:	Kommentar:
`00 { 115-Byte Prgm }` `01 LBL "SSS"`	Zeile 01: Definition des Programmanfangs.

`02 INPUT "S1"`
`03 INPUT "S2"`
`04 INPUT "S3"`
	Zeile 02 – 04: Eingabeaufforderung für die Werte der drei Seiten und Speichern der Werte in Variablen.

`05 RCL "S1"`
`06 RCL+ "S2"`
`07 RCL+ "S3"`
`08 2`
	Zeile 05 – 40: Berechnung von W_1, W_2 und W_3. Speichern der Werte in benannten Variablen.

```
09 ÷
10 STO "P"
11 X↑2
12 LASTX
13 RCL× "S2"
14 -
15 RCL "S1"
16 RCL× "S3"
17 ÷
18 SQRT
19 ACOS
20 2
21 ×
22 STO "W3"
23 SIN
24 RCL "P"
25 X↑2
26 LASTX
27 RCL× "S1"
28 -
29 RCL÷ "S2"
30 RCL÷ "S3"
31 SQRT
32 ACOS
33 2
34 ×
35 STO "W2"
36 RCL+ "W3"
37 COS
38 +/-
39 ACOS
40 STO "W1"
```

```
41 SF 21
42 VIEW "W1"
43 VIEW "W2"
44 VIEW "W3"
```

Zeile 41 – 44: Anzeigen (oder Ausdrucken) der berechneten Ergebnisse.

```
45 END
```

Zeile 45: Abschluß des Programms.

Ausführen des Programms

Sie können SSS durch eine der nachstehenden Tastenfolgen ausführen.

Verwenden des Programmkatalogs. Das globale Label SSS wurde bei der Eingabe von Programmzeile 01 automatisch im Programmkatalog aufgenommen. Sie können das Programm durch Drücken von

■ [CATALOG] PGM SSS

ausführen. Diese Folge erfordert wenigstens vier Tasten, in Abhängigkeit davon, wo sich das Label SSS im Programmkatalog befindet. (Wenn Sie nach SSS noch mehr als fünf Programme erzeugt haben, ist die Taste [▼] zum Auffinden des Labels SSS zu verwenden.)

Verwenden von XEQ. Wenn Sie [XEQ] drücken, wird das Programmkatalog-Menü automatisch angezeigt. Sie können daher SSS durch Drücken von

[XEQ] SSS

ausführen. Diese Folge erfordert wenigstens zwei Tasten, abhängig davon, wo sich SSS im Programmkatalog befindet.

Anwenden des CUSTOM Menüs. Alternativ können Sie SSS dem CUSTOM Menü zuweisen, indem Sie zuerst die Tastenfolge

■ [ASSIGN] PGM SSS

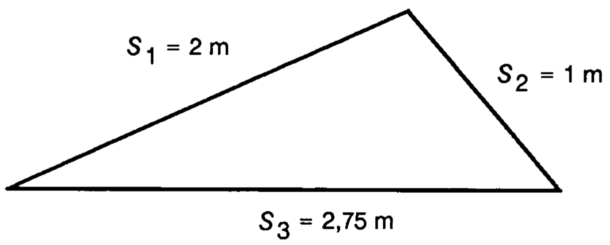
ausführen und anschließend die gewünschte Menütaste drücken.

Das Programm kann nun direkt vom CUSTOM Menü aus durch Drücken von

■[CUSTOM] SSS

gestartet werden. Diese Folge erfordert drei Tasten, wenn Sie zuerst das CUSTOM Menü auswählen, und nur eine Taste für aufeinanderfolgende Ausführungen, sofern Sie in der momentanen Menüzeile bleiben.

Beispiel: Ausführung eines Programms vom CUSTOM Menü. Berechnen Sie die Winkel (in Grad) des nachstehenden Dreiecks.



Weisen Sie SSS dem CUSTOM Menü zu. Stellen Sie "Grad" als Winkelmodus ein. Führen Sie PRON aus, falls Sie über einen Drucker verfügen und die Ergebnisse ausdrucken möchten. Starten Sie nun das Programm.

■[ASSIGN] PGM
SSS
■[MODES] DEG
(■[PRINT] [▲] PON)
■[CUSTOM] SSS

```
S1?0,0000
SSS
```

Geben Sie die Werte für S_1 ein und setzen Sie das Programm fort.

2 [R/S]

```
S2?0,0000
SSS
```


Geben Sie den Wert für S_2 und anschließend den für S_3 ein. Das Programm berechnet nun die drei Winkel und zeigt *W1*, das erste Ergebnis, an. (Wenn Sie PRON zum Ausdrucken der Ergebnisse ausgeführt haben, werden die nächsten zwei Anzeigen nicht angezeigt.)

1 [R/S] 2,75 [R/S]

```
W1=129,8384
SSS
```

Setzen Sie das Programm fort, um W_2 anzuzeigen.

[R/S]

```
W2=33,9479
SSS
```

Setzen Sie das Programm fort, um W_3 anzuzeigen.

[R/S]

```
W3=16,2136
SSS
```

Beenden Sie das Programm.

[EXIT]

```
y: 0,2792
x: 129,8384
```

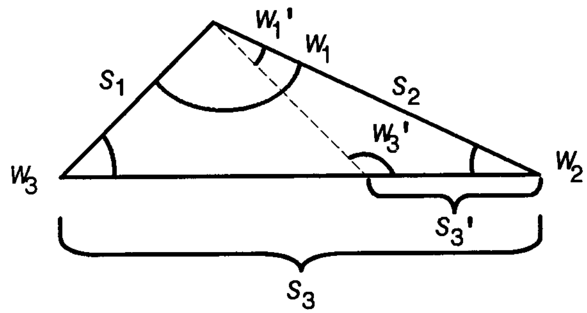
Verzweigungen

Eine Verzweigungsanweisung erlaubt Ihnen, die Programmausführung an einer anderen Stelle des Programms (statt der nächsten Anweisung) fortzusetzen. Die Verzweigung kann dabei folgender Art sein:

- Bedingt (abhängig von einem Test).
- Unbedingt (typisch zum Aufrufen einer Subroutine, wobei nach deren Abschluß die Programmsteuerung wieder an das Hauptprogramm zurückgegeben wird).
- Menügesteuert (von Ihnen über ein programmiertes Menü ausgewählt).

Bedingtes Verzweigen

Das Programm SSW auf Seite 24 und 25 in diesem Abschnitt veranschaulicht die Anwendung von bedingten Verzweigungen. SSW berechnet die zwei unbekannten Winkel und die unbekannte Seite eines Dreiecks, wenn zwei Seiten und der angrenzende Winkel bekannt sind (S_1, S_2 und W_2).



Die Gleichungen zum Berechnen von W_3, W_1, und S_3 lauten:

$$W_3 = \text{arc sin}\left[\left(\frac{S_2}{S_1}\right)\sin W_2\right]$$

$$W_1 = \text{arc cos}\,[\,-\cos(W_2 + W_3)\,]$$

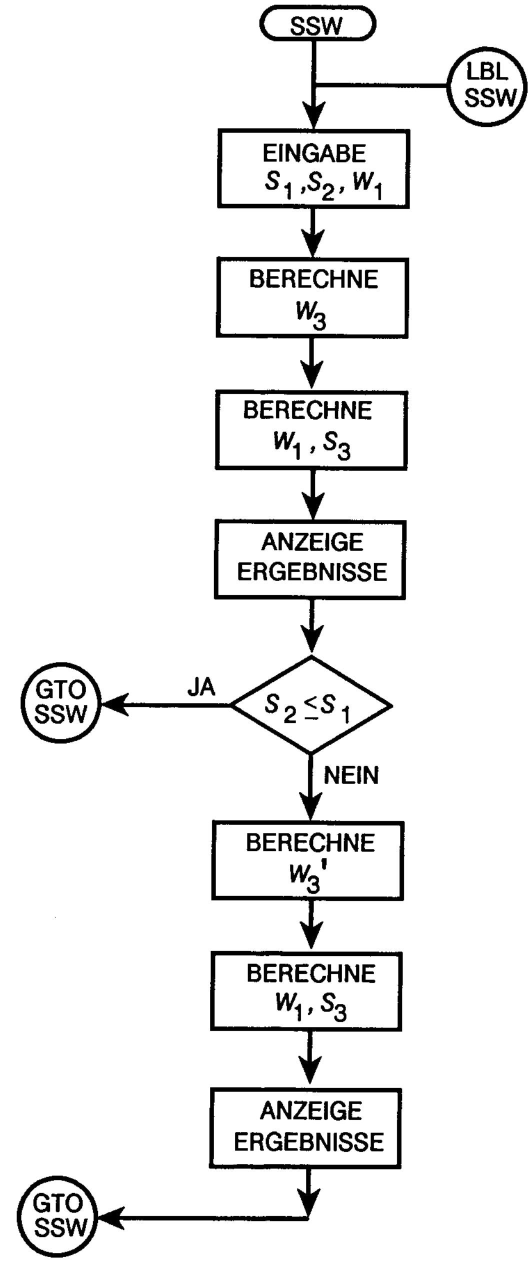
$$S_3 = S_1 \cos W_3 + S_2 \cos W_2$$

Aus der Abbildung geht hervor, daß es zwei mögliche Lösungen gibt, wenn S_2 größer als S_1 und W_3 ungleich 90° ist. Dies führt zu einer vierten Gleichung:

$$W_3{}' = \text{arc cos}\,(\,-\cos W_3)$$

SSW berechnet beide mögliche Ergebnissätze.

Nachstehend finden Sie das Flußdiagramm für das Programm.



Beachten Sie im Flußdiagramm, daß das Programm den ersten Ergebnissatz berechnet und danach die Werte von S_1 und S_2 vergleicht. Abhängig vom Ausgang des Vergleichs kehrt das Programm entweder zum Label SSW zurück oder berechnet den zweiten Ergebnissatz. Das Programm benutzt hier eine *bedingte Verzweigung*. Die erforderliche Tastenfolge ist in der nachstehenden Auflistung hervorgehoben. (Diese bedingte Verzweigung basiert auf dem Vergleich *zweier Zahlen*. Später in diesem Kapitel wird auf Verzweigungen, basierend auf der Abfrage von *Flags*, eingegangen.)

Um SSW einzutippen: Erzeugen Sie zuerst die Variablen *S1*, *S2*, *S3*, *W1*, *W2* und *W3*, bevor Sie das Programm eintippen. (Diese Variablen existieren bereits, wenn Sie SSS gespeichert haben.)

Programm:	**Kommentar:**
00 { 157-Byte Prgm } 01 LBL "SSW"	
02 SF 21	
03 INPUT "S1" 04 INPUT "S2" 05 INPUT "W2"	Zeile 03–05: Eingeben der bekannten Variablen.
06 SIN 07 RCL× "S2" 08 RCL÷ "S1" 09 ASIN 10 STO "W3" 11 RCL+ "W2" 12 COS 13 +/- 14 ACOS 15 STO "W1" 16 RCL "W2" 17 COS 18 RCL× "S2" 19 RCL "W3" 20 COS 21 RCL× "S1" 22 + 23 STO "S3"	Zeile 06–23: Berechnen der unbekannten Variablen.
24 VIEW "W1" 25 VIEW "S3" 26 VIEW "W3"	Zeile 24–26: Anzeigen (oder Drucken) der unbekannten Variablen.
27 RCL "S1" 28 RCL "S2" 29 X≤Y? 30 GTO "SSW"	Zeile 27–30: Vergleichen, ob S_2 kleiner oder gleich S_1 ist. Falls ja, Rücksprung zum Programmanfang, ansonsten Berechnung des zweiten Ergebnissatzes.
31 RCL "W3" 32 COS 33 +/- 34 ACOS 35 STO "W3" 36 RCL+ "W2" 37 COS 38 +/- 39 ACOS 40 STO "W1" 41 RCL "W2" 42 COS 43 RCL× "S2" 44 RCL "W3" 45 COS 46 RCL× "S1" 47 + 48 STO "S3"	Zeile 31–48: Berechnen des zweiten Ergebnissatzes.
49 VIEW "W1" 50 VIEW "S3" 51 VIEW "W3" 52 GTO "SSW"	Zeile 49–52: Anzeigen des zweiten Ergebnissatzes und Rücksprung zum Programmanfang.
53 END	

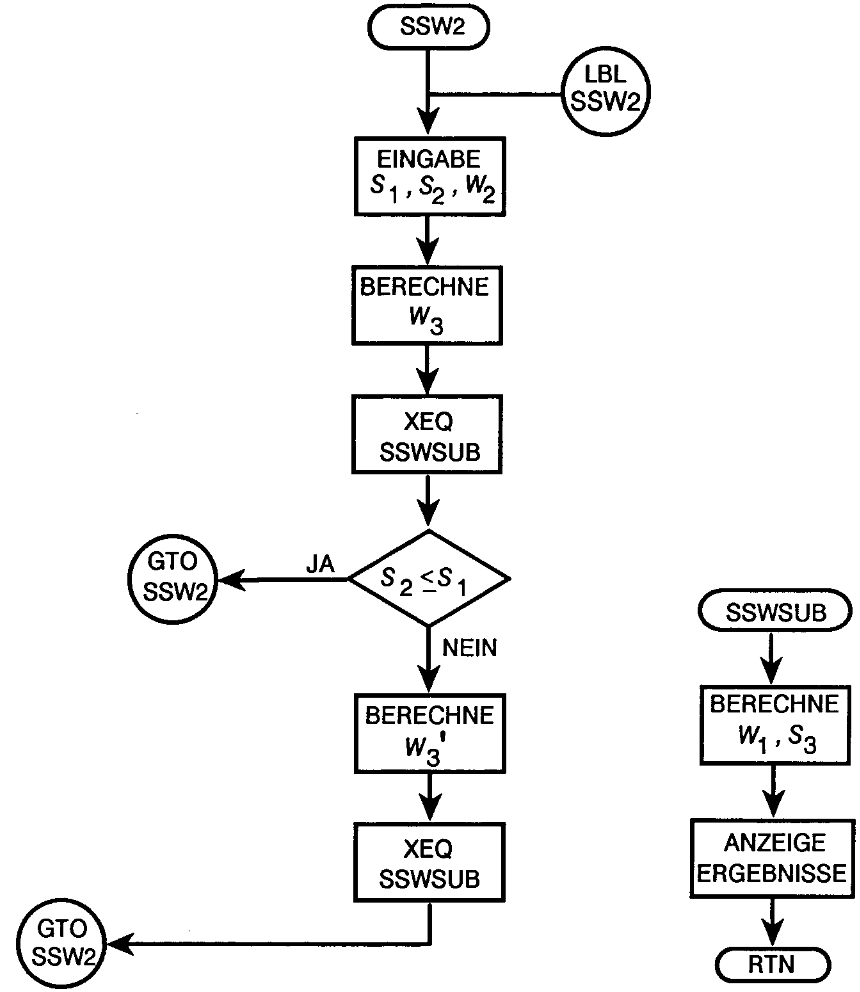
Subroutinen

Eine Routine besteht aus einer Reihe von Anweisungen, die mit einem lokalen oder globalen Label beginnen und mit einer RTN oder END Anweisung abgeschlossen sind. (Die Programme SSS und SSW sind Routinen.) Eine Routine wird zur *Subroutine* (Unterprogramm), wenn sie von einer anderen Routine über eine XEQ Anweisung *aufgerufen* (ausgeführt) wird. Nachdem die Subroutine ausgeführt wurde, wird die Programmsteuerung aufgrund der RTN oder END Anweisung am Ende der Subroutine wieder an die *Hauptroutine zurückgegeben*.

Beachten Sie, daß SSW den zweiten Ergebnissatz (sofern einer existiert) berechnet, indem zuerst W_3' berechnet wird. Danach werden die verbleibenden Unbekannten unter Verwendung der *selben* Gleichungen, die zur Berechnung des ersten Satzes angewendet wurden, berechnet. Anschließend erfolgt die Anzeige der Ergebnisse unter Verwendung der gleichen Anweisungen, welche zum Anzeigen des ersten Satzes benutzt wurden. Durch das Zusammenfassen dieser gemeinsamen Anweisungen in einer Subroutine erreichen Sie programmtechnisch folgende Vorteile:

- Das Programm wird kürzer.
- Es wird übersichtlicher und einfacher zu lesen.
- Es ist einfacher zu schreiben.
- Modifikationen lassen sich leichter vornehmen.

Nachstehend ein Flußdiagramm für das neue SSW2 Programm, welches eine Subroutine verwendet.



Um SSW2 einzutippen:

1. Erzeugen Sie vorab die Variablen *S1*, *S2*, *S3*, *W1*, *W2* und *W3*.
2. Erzeugen Sie das Label SSWSUB, wenn Sie mit dem Eintippen des Programms beginnen.

Programm: / **Kommentar:**

```
00 { 137-Byte Prgm }
01 LBL "SSW2"
02 SF 21
03 INPUT "S1"
04 INPUT "S2"
05 INPUT "W2"
```

```
06 SIN
07 RCL× "S2"
08 RCL÷ "S1"
09 ASIN
```

Zeile 06–09: Berechne W_3.

```
10 XEQ "SSWSUB"
```

Zeile 10: Aufruf von Subroutine SSWSUB zur Berechnung von W_1 und S_3. Diese unbedingte Verzweigung benutzt eine XEQ Anweisung; die nächste RTN (oder END) Anweisung bringt die Programmausführung zurück zu Zeile 11. (Folgen Sie nun dem Sprung zu Zeile 21.)

```
11 RCL "S1"
12 RCL "S2"
13 X≤Y?
14 GTO "SSW2"
```

Zeile 11–14: Wenn S_2 kleiner oder gleich S_1 ist, erfolgt ein Rücksprung zum Programmanfang, ansonsten wird der zweite Ergebnissatz berechnet.

```
15 RCL "W3"
16 COS
17 +/-
18 ACOS
```

Zeile 15–18: Berechne $W_3{}'$.

```
19 XEQ "SSWSUB"
20 GTO "SSW2"
```

Zeile 19–20: Aufruf von Subroutine SSWSUB zur Berechnung von $W_1{}'$ und $S_3{}'$, anschließend Rücksprung zum Programmanfang.

```
21 LBL "SSWSUB"
22 STO "W3"
23 RCL+ "W2"
24 COS
25 +/-
26 ACOS
27 STO "W1"
28 RCL "W2"
29 COS
30 RCL× "S2"
31 RCL "W3"
32 COS
33 RCL× "S1"
34 +
35 STO "S3"
36 VIEW "W1"
37 VIEW "S3"
38 VIEW "W3"
39 RTN
```

Subroutine SSWSUB, Zeile 21–39: Berechne die Werte von W_1 und S_3 ($W_1{}'$ und $S_3{}'$ im zweiten Ergebnissatz) und zeige die Ergebnisse an.

```
40 END
```


SSW2 ist um 13 Zeilen bzw. 20 Bytes kürzer als SSW.

Verschachtelte Subroutinen. Im folgenden Abschnitt sind alle 5 möglichen Dreiecksberechnungen/-lösungen über die Subroutinen A bis E realisiert. Beachten Sie im Flußdiagramm für 3ECK (Seiten 30–31), daß Subroutine B, welche zur Lösung der ursprünglichen SSW Bedingung dient, selbst wiederum Subroutine SSWSUB zur Berechnung von W_2 und S_3 aufruft. In 3ECK ist Subroutine SSWSUB *geschachtelt* in Subroutine B. Beim Aufruf von Subroutine SSWSUB durch Subroutine B gibt es zwei *ausstehende* Subroutinen. Der HP-42S kann bis zu 8 ausstehende Subroutinen verarbeiten.

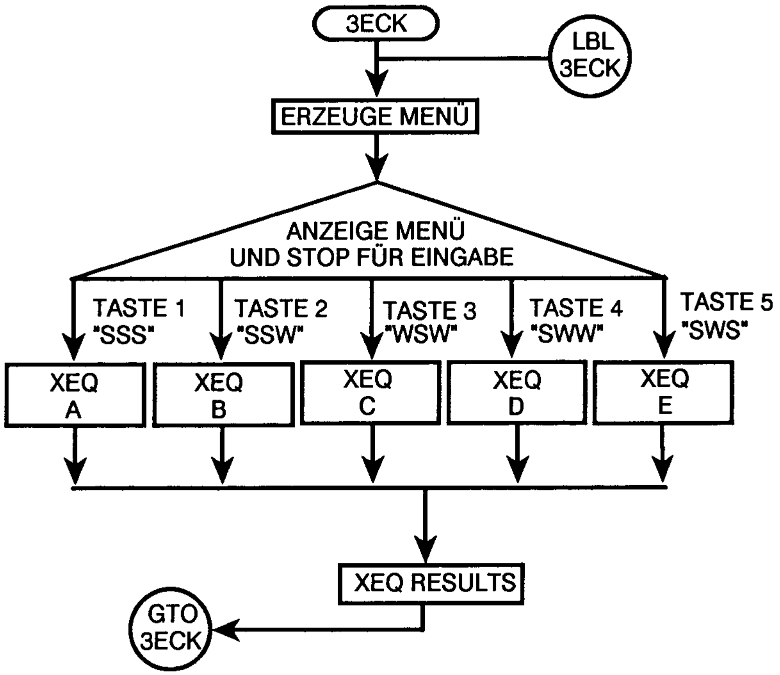
Menügesteuerte Verzweigungen

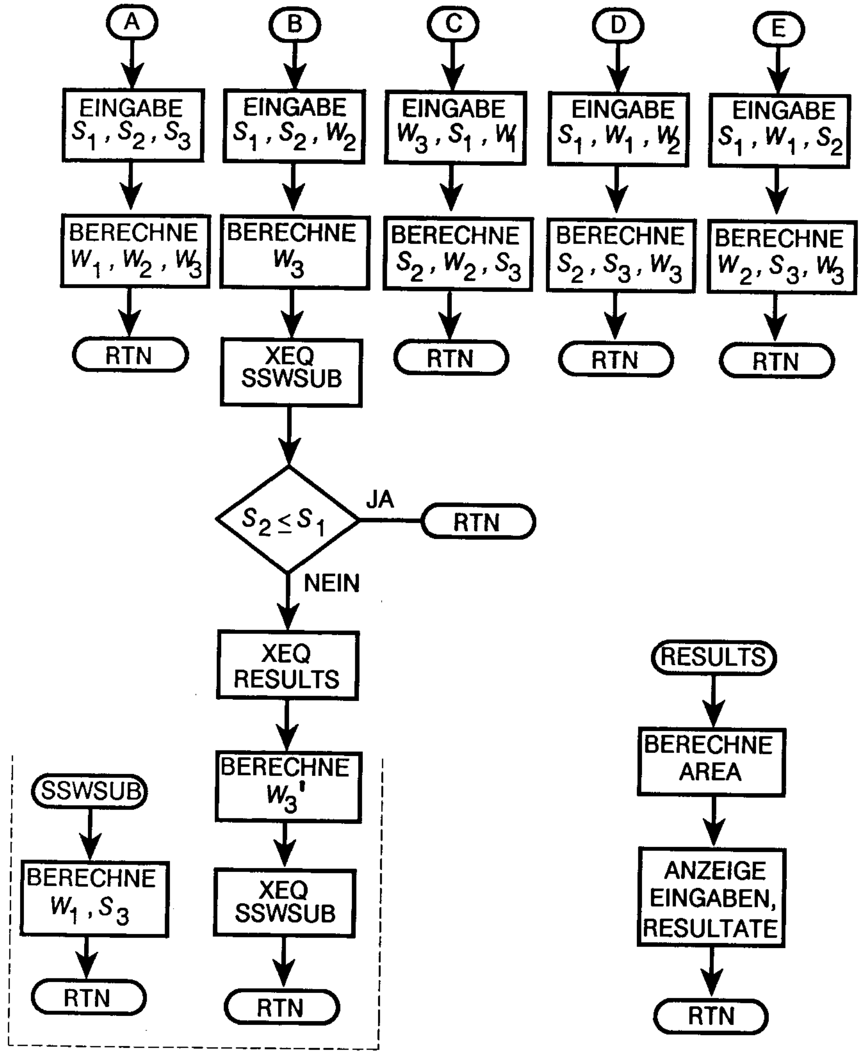
Programmierbare Menüs erlauben *Ihnen* das Treffen einer Entscheidung während des Programmablaufs, indem benannte Menüfelder angezeigt werden, die das Verzweigen zu bestimmten Stellen im Programmspeicher bewirken. Durch Verwenden von KEY XEQ oder KEY GTO Anweisungen (die wie XEQ und GTO wirken) kann jedes Label im Programmspeicher als programmierbare Menütaste spezifiziert werden. Wird eine MENU und STOP Anweisung unmittelbar nacheinander ausgeführt, so wird das Programm unterbrochen, das programmierbare Menü angezeigt und die Tasten 1 bis 9 (die obersten 6 Tasten sowie ▲, ▼ und EXIT) übernehmen ihre korrespondierenden Menüfunktionen.

Die vorangehenden zwei Programme, SSS und SSW, berechneten jeweils eine der fünf möglichen Dreiecklösungen. Entsprechend enthalten die anderen Lösungen:

- S_2, W_2 und S_3 (wobei W_3, S_1, und W_1 bekannt sind).
- S_2, S_3 und W_3 (wobei S_1, W_1, und W_2 bekannt sind).
- W_2, S_3 und W_3 (wobei S_1, W_1, und S_2 bekannt sind).

Nachstehend finden Sie das Flußdiagramm für das Programm "3ECK". Dieses Programm enthält jede der fünf Lösungen in einer Subroutine und erzeugt ein programmierbares Menü, über welches Sie durch Drücken einer Menütaste die gewünschte Subroutine starten können.





Das Dreiecksymbol im Flußdiagramm zeigt an, an welcher Stelle das Programm anhält, um das Menü anzuzeigen. Sie wählen die gewünschte Lösung, indem Sie die zugehörige Menütaste drücken.

Nachstehend die erforderlichen Programmzeilen.

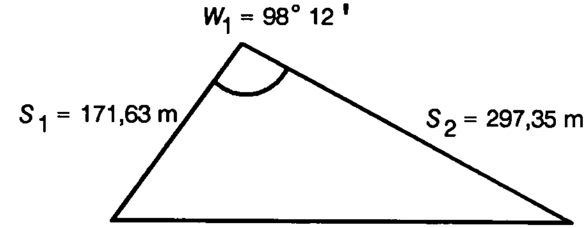
Programm:	**Kommentar:**

`03 "SSS"`
`04 KEY 1 XEQ A`
`05 "SSW"`
`06 KEY 2 XEQ B`
`07 "WSW"`
`08 KEY 3 XEQ C`
`09 "SWW"`
`10 KEY 4 XEQ D`
`11 "SWS"`
`12 KEY 5 XEQ E`
	Zeile 03–12: Erzeuge die Menütasten. (Z.B. benennt Zeile 03 die 1. Menütaste als "SSS" und Zeile 04 definiert, daß diese Taste Subroutine A ausführt.)

`13 MENU`
`14 STOP`
`15 XEQ "RESULTS"`
`16 GTO "3ECK"`
	Zeile 13–16: Wähle das Menü (Zeile 13) und unterbreche die Programmausführung (Zeile 14). (Das Menü wird angezeigt, wenn das Programm unterbrochen wird.) Rufe nach der Ausführung einer der Subroutinen A bis E die Subroutine RESULTS auf, um die Ergebnisse anzuzeigen (Zeile 15). Rücksprung zum Label 3ECK bzw. zum Programmanfang.

Die vollständige Auflistung von 3ECK finden Sie auf Seite 60–65 am Ende dieses Kapitels.

Beispiel: Ein programmierbares Menü. Ein Vermessungsbüro ist mit der Vermessung eines dreiecksförmigen Grundstücks beauftragt. Von Punkt A wird die Entfernung zu Punkt B und C gemessen, anschließend erfolgt das Messen des Winkels zwischen AB und AC.



Dieses Beispiel behandelt eine SWS (Seite-Winkel-Seite) Aufgabenstellung.

Wählen Sie Grad als Winkelmodus. (Führen Sie PRON aus, falls die Ergebnisse gedruckt werden sollen.) Beginnen Sie nun mit der Programmausführung.

Tasten	Anzeige
■[MODES] DEG (■[PRINT] [▲] PON) [XEQ] 3ECK	`x: 0,0000` `SSS SSW WSW SWW SWS`

Wählen Sie die SWS Routine durch Drücken von Menütaste 5.

Tasten	Anzeige
SWS	`S1?0,0000` `SSS SSW WSW SWW SWS`

Tippen Sie den Wert für S_1 ein und setzen Sie das Programm fort.

Tasten	Anzeige
171,63 [R/S]	`W1?0,0000` `SSS SSW WSW SWW SWS`

Tippen Sie die Werte für W_1 und S_2 ein (Sie müssen W_1 in den äquivalenten Dezimalwert umrechnen). Das Programm berechnet die Unbekannten und zeigt die Eingabewerte sowie die Ergebnisse an.

Tasten	Anzeige
98,12 ■[CONVERT] →HR [R/S] 297,35 [R/S]	`S1=171,6300` `x: 25.256,2094`

Drücken Sie dreimal [R/S], um W_2 anzusehen.

Tasten	Anzeige
[R/S] [R/S] [R/S]	`W2=27,8270` `x: 25.256,2094`

Drücken Sie erneut [R/S], um S_3 anzusehen.

[R/S]

```
S3=363,9118
x: 25.256,2094
```

Drücken Sie erneut [R/S], um W_3 anzusehen.

[R/S]

```
W3=53,9730
x: 25.256,2094
```

Drücken Sie nochmals [R/S], um *AREA* anzusehen.

[R/S]

```
AREA=25.256,2094
x: 25.256,2094
```

Wenn Sie erneut [R/S] drücken, erhalten Sie das Menü angezeigt.

[R/S]

```
x: 25.256,2094
SSS  SSW  WSW  SWW  SWS
```

Beenden Sie das Programm.

[EXIT]

```
y: 27,8270
x: 25.256,2094
```

Mehrzeilige Menüs. Das vorangehende Programm 3ECK erzeugt Menüfelder für fünf der sechs obersten Tasten und weist jeder benannten Taste eine KEY XEQ Anweisung zu.

Ein mehrzeiliges Menü hat *mehr als nur eine Zeile* mit benannten Menüfeldern. (Z.B. verfügt das CLEAR Menü über zwei Zeilen.) Wenn Sie ein mehrzeiliges Menü eingeben, können Sie mit Hilfe der Tasten [▼] und [▲] zu den einzelnen Zeilen des Menüs übergehen. (Als Indikator erscheint der ▼▲ Indikator in der Anzeige.)

Sie können ein mehrzeiliges Menü in einem Programm *emulieren*, indem Sie eine KEY GTO Anweisung Menütaste 7 (der [▲] Taste) und Menütaste 8 (der [▼] Taste) zuweisen. (KEY GTO oder KEY XEQ Anweisungen für Menütaste 7 und 8 schalten auch *automatisch* den ▼▲ Indikator in der Anzeige ein.)

Beachten Sie das nachstehende einfache Menü mit Rechnerfunktionen.

```
| √x
| LOG
| LN
▼▲
| R↓
| SIN
| COS
▼▲
| X<>Y
| +/-
```

Nachstehend finden Sie ein Programm, welches obiges mehrzeiliges Menü emuliert.

Um ZEIL1 einzutippen:

1. Erzeugen Sie die Labels ZEIL1, ZEIL2 und ZEIL3, bevor Sie mit der Programmeingabe beginnen.
2. Beachten Sie, daß die Programmzeilen 03, 05, 07, 16, 18, 20, 29 und 31 Alpha-Strings darstellen.

Programm:	**Kommentar:**
`00 { 196-Byte Prgm }`	
`01 LBL "ZEIL1"`	
`02 CLMENU`	Zeile 01–13: Lösche die momentanen Menüdefinitionen, erzeuge danach die erste Menüzeile und zeige diese an. Zuweisung von Verzweigungen zu den Tasten 7 und 8 ([▲] und [▼]) zur vorangehenden bzw. nachfolgenden Zeile (Zeile 09–10).
`03 "√"`	
`04 KEY 3 XEQ 01`	
`05 "LOG"`	
`06 KEY 4 XEQ 02`	
`07 "LN"`	
`08 KEY 5 XEQ 03`	
`09 KEY 7 GTO "ZEIL3"`	
`10 KEY 8 GTO "ZEIL2"`	
`11 MENU`	
`12 STOP`	
`13 GTO "ZEIL1"`	

```
14 LBL "ZEIL2"
15 CLMENU
16 "R↓"
17 KEY 3 XEQ 04
18 "SIN"
19 KEY 4 XEQ 05
20 "COS"
21 KEY 5 XEQ 06
22 KEY 7 GTO "ZEIL1"
23 KEY 8 GTO "ZEIL3"
24 MENU
25 STOP
26 GTO "ZEIL2"
```

Zeile 14–26: Lösche die momentanen Menüdefinitionen, erzeuge die zweite Menüzeile und zeige diese an. Zuweisung von Verzweigungen zu den Tasten 7 und 8 zur vorangehenden bzw. nachfolgenden Zeile (Zeile 22–23).

```
27 LBL "ZEIL3"
28 CLMENU
29 "X<>Y"
30 KEY 3 XEQ 07
31 "+/-"
32 KEY 4 XEQ 08
33 KEY 7 GTO "ZEIL2"
34 KEY 8 GTO "ZEIL1"
35 MENU
36 STOP
37 GTO "ZEIL3"
```

Zeile 27–37: Lösche die momentanen Menüdefinitionen, erzeuge die dritte Menüzeile und zeige diese an. Zuweisung von Verzweigungen zu den Tasten 7 und 8 zur vorangehenden bzw. nachfolgenden Zeile (Zeile 33–34).

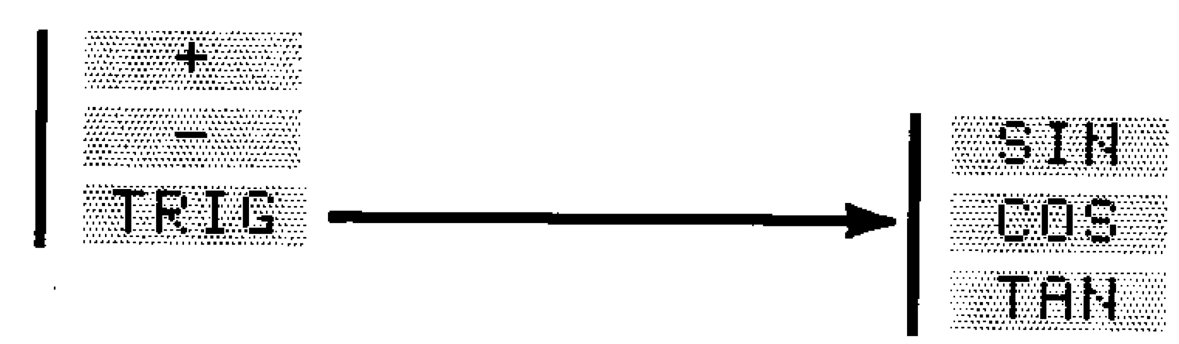
```
38 LBL 01
39 SQRT
40 RTN
41 LBL 02
42 LOG
43 RTN
44 LBL 03
45 LN
46 RTN
47 LBL 04
48 R↓
49 RTN
50 LBL 05
51 SIN
52 RTN
53 LBL 06
54 COS
55 RTN
56 LBL 07
57 X<>Y
58 RTN
59 LBL 08
60 +/-
61 RTN

62 END
```

Subroutine 01–08, Zeile 38–61: Führe die entsprechenden Rechnerfunktionen der einzelnen Menüfelder aus.

Verschachtelte Menüs. In vielen Menüs erzeugt eine oder mehrere der sechs obersten Menütasten ein neues Menü, welches als *verschachteltes Menü* oder *Untermenü* bezeichnet wird. So führt z.B. im PGM.FCN Menü das Drücken der Menütaste X?0 zur Anzeige eines verschachtelten Menüs mit ähnlichen Funktionen (X=0?, X≠0?, ..., X≥0?). Um wieder zum Hauptmenü zurückzukehren, ist [EXIT] zu drücken.

Sie können ein verschachteltes Menü in einem Programm *emulieren*, indem Sie eine KEY GTO Anweisung einer beliebigen Menütaste der obersten Tastenreihe zuweisen. Beachten Sie nachstehendes einfaches Menü von Rechnerfunktionen:



Hier nun ein Programm, welches die vorangehende Menüstruktur emuliert.

Um EBE1 einzutippen:

1. Erzeugen Sie die Labels EBE1 und EBE2 vor Beginn der Programmeingabe.
2. Beachten Sie, daß Zeile 03, 05, 07, 14, 16 und 18 Alpha-Strings darstellen.

Programm: **Kommentar:**

```
00 { 108-Byte Prgm }
01 LBL "EBE1"
```

```
02 CLMENU
03 "+"
04 KEY 2 XEQ 01
05 "-"
06 KEY 3 XEQ 02
07 "TRIG"
08 KEY 5 GTO "EBE2"
09 MENU
10 STOP
11 GTO "EBE1"
```

Zeile 01 – 11: Erzeuge die erste Ebene des Menüs und zeige diese an. Zuweisung für Taste 3 (mit TRIG bezeichnet) zum Verzweigen zu Label EBE2, um das verschachtelte Menü zu erzeugen (Zeile 08).

```
12 LBL "EBE2"
13 CLMENU
14 "SIN"
15 KEY 4 XEQ 11
16 "COS"
17 KEY 5 XEQ 12
18 "TAN"
19 KEY 6 XEQ 13
20 KEY 9 GTO "EBE1"
21 MENU
22 STOP
23 GTO "EBE2"
```

Zeile 12 – 23: Erzeuge das verschachtelte Menü und zeige dieses an. Zuweisung für Taste 9 ([EXIT] Taste) zur Rückkehr zu Label EBE1 (Zeile 20).

```
24 LBL 01
25 +
26 RTN
27 LBL 02
28 -
29 RTN
30 LBL 11
31 SIN
32 RTN
33 LBL 12
34 COS
35 RTN
36 LBL 13
37 TAN
38 RTN
```

Subroutine 01, 02 und 11 – 13, Zeile 24 – 38: Führe die entsprechenden Rechnerfunktionen der jeweiligen Menüfelder aus.

```
39 END
```


Bestimmte Schleifen

Eine Schleife, die eine spezifizierte Anzahl oft durchlaufen wird, wird als *bestimmte Schleife* bezeichnet. Sie können eine bestimmte Schleife mit Hilfe eines lokalen oder globalen Labels, einer ISG oder DSE Anweisung, oder einer GTO Anweisung erzeugen.

Das Programm WEG im folgenden Abschnitt verwendet eine bestimmte Schleife zur Berechnung des zurückgelegten Weges eines Objekts, welches sich mit einer gleichförmigen Geschwindigkeit bewegt.

Die Gleichung für eine gleichförmige Bewegung auf einer glatten Oberfläche lautet:

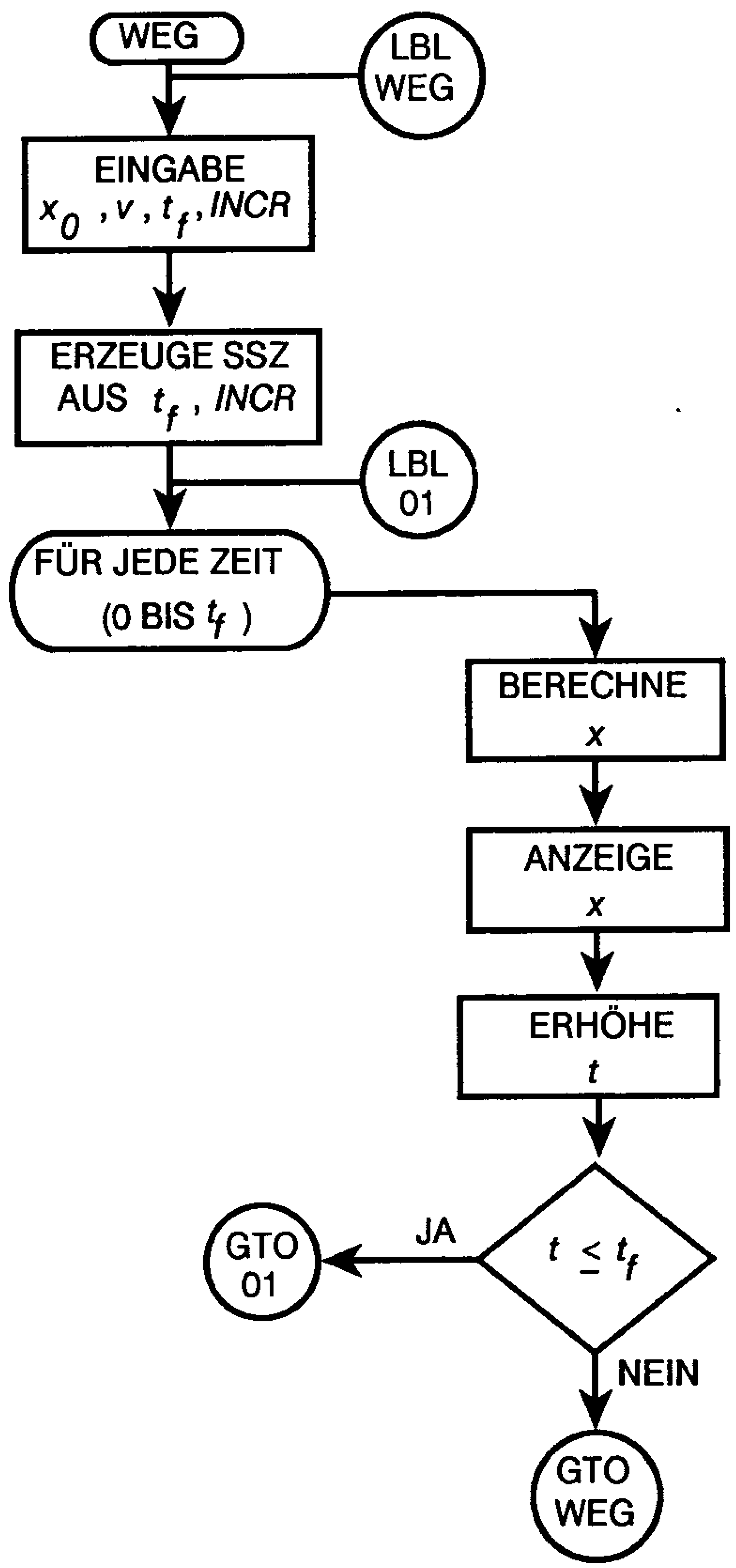
$$x = x_0 + vt$$

wobei:

x = gesamte Strecke
x_0 = Ausgangsposition
v = Geschwindigkeit
t = benötigte Zeit

WEG berechnet den zurückgelegten Weg nach aufeinanderfolgenden Zeitintervallen, von $t = 0$ bis $t = t_f$. Es erzeugt eine Schleifensteuerzahl mit dem Format .fffii, indem Sie zur Eingabe eines Wertes für t_f und für *INCR* (Increment, der Wert des Zeitintervalls) aufgefordert werden. t_f wird zum fff-Teil der Schleifensteuerzahl, während *INCR* den ii-Teil darstellt.

Nachstehend finden Sie das Flußdiagramm für WEG.



Um WEG einzutippen: Erzeugen Sie zuerst die Variablen *x*, *x0*, *v*, *tF*, *INCR*, *fff*, *ii* und *SSZ*, bevor Sie mit der Programmeingabe beginnen.

Programm:	**Kommentar:**
`00 { 100-Byte Prgm }` `01 LBL "WEG"`	
`02 SF 21`	

`03 INPUT "x0"`
`04 INPUT "v"`
`05 INPUT "tF"`
`06 INPUT "INCR"`
`07 RCL "tF"`
`08 1E-3`
`09 ×`
`10 STO "fff"`
`11 RCL "INCR"`
`12 1E-5`
`13 ×`
`14 STO "ii"`
`15 RCL+ "fff"`
`16 STO "SSZ"`
	Zeile 03 – 16: Eingabeaufforderung für Variablen. Erzeugen der Schleifensteuerzahl.

`17 LBL 01`
`18 RCL "SSZ"`
`19 IP`
`20 RCL× "v"`
`21 RCL+ "x0"`
`22 STO "x"`
`23 CLX`
`24 VIEW "x"`
`25 ISG "SSZ"`
`26 GTO 01`
`27 GTO "WEG"`
	Zeile 17 – 27: Berechne über eine bestimmte Schleife aufeinanderfolgende Werte von *x*. (Beachten Sie, daß der ganzzahlige Teil von *SSZ* in Zeile 19 die Zeit *t* darstellt.)

`28 END`	

Beispiel: Schleifensteuerung in einem Programm. Berechnen Sie die aufeinanderfolgenden Werte für den zurückgelegten Weg x eines Objekts, wobei ein Intervall von 5 Sekunden für die Zeit $t = 0$ bis $t = 15$ Sekunden sowie $x_0 = 10$ m und $v = 20$ m/s zu verwenden ist.

Start des Programms.

[XEQ] WEG

```
Y: 0,0000
x0?0,0000
```

Eingabe der Werte für x_0 und v.

10 [R/S] 20 [R/S]

```
Y: 20,0000
tF?0,0000
```

Eingabe der Werte für t_f und Programmfortsetzung.

15 [R/S]

```
Y: 15,0000
INCR?0,0000
```

Eingabe des Wertes von *INCR* (Zeitintervall) und Programmfortsetzung.

5 [R/S]

```
x=10,0000
x: 0,0000
```

Der Wert von x bei $t = 0$ ist 10. Drücken Sie nochmals [R/S], um den Wert von x bei $t = 5$ anzuzeigen.

[R/S]

```
x=110,0000
x: 0,0000
```

Drücken Sie [R/S], um den Wert von x bei $t = 10$ anzuzeigen.

[R/S]

```
x=210,0000
x: 0,0000
```

Drücken Sie erneut [R/S], um den Wert von x bei $t = 15$ anzuzeigen.

[R/S]

```
x=310,0000
x: 0,0000
```

Drücken Sie [R/S] erneut, wodurch Sie zur Eingabe neuer Variableninhalte aufgefordert werden. Beenden Sie danach das Programm.

[R/S] [EXIT]

```
Y: 0,0000
x: 10,0000
```


Indirekte Adressierung

Indirekte Adressierung ist ein nützliches Mittel bei der Programmierung, insbesondere bei der Steuerung einer Schleife. Im Verzeichnis der Operationen des Benutzerhandbuchs sind alle Funktionen gekennzeichnet, welche sich für die indirekte Adressierung einsetzen lassen. In diesem Abschnitt werden Ihnen drei Anwendungsmöglichkeiten der indirekten Adressierung vorgestellt.

Indirekte Adressierung zum Initialisieren von Datenspeicherregister. Das Programm INIT fordert zur Eingabe von Daten auf und speichert diese in aufeinanderfolgenden Registern, wobei INPUT IND in einer bestimmten Schleife verwendet wird. Dies stellt eine hilfreiche Initialisierungsroutine dar, wenn Sie Register anstatt von Variablen für die Datenspeicherung verwenden.

```
00 { 31-Byte Prgm }
01 LBL "INIT"
```

```
02 1,01
03 STO "SSZ"
```

Zeile 02 – 03: Erzeuge eine Schleifensteuerzahl und speichere sie in *SSZ*. Diese Variable hat einen Anfangswert von 1, einen Endwert von 10 und den Standardwert 1 für die Schrittweite (Increment).

```
04 LBL 01
05 INPUT IND "SSZ"
06 ISG "SSZ"
07 GTO 01
```

Zeile 04 – 07: Eingabeaufforderung für Daten für die aufeinanderfolgenden Register $R_{01} - R_{10}$.

```
08 END
```

Indirekte Adressierung zum Löschen von Registern. Die nachstehende Routine löscht eine spezifizierte Anzahl von Registern unter Verwendung von STO IND in einer bestimmten Schleife.

Programm: / **Kommentar:**

```
00 { 79-Byte Prgm }
01 LBL "CLEAR"
```

```
02 0
```

Zeile 02: Initialisiere das X-Register mit 0.

```
03 "ERSTES REG?"
04 PROMPT
05 STO "SSZ"
06 "LETZTES REG?"
07 PROMPT
08 1E-3
09 ×
10 STO+ "SSZ"
```

Zeile 03 – 10: Erzeuge eine Schleifensteuerzahl in *SSZ*. Der Startwert entspricht dem ersten zu löschenden Speicherregister, der Endwert entspricht dem letzten zu löschenden Speicherregister; als Schrittweite wird 1 verwendet.

```
11 LBL 10
12 0
13 STO IND "SSZ"
14 ISG "SSZ"
15 GTO 10
```

Zeile 11 – 15: Setze den Inhalt des spezifizierten Registerblocks nacheinander auf Null.

```
16 TONE 9
17 "FERTIG"
18 PROMPT
```

Zeile 16 – 18: Gib ein Tonsignal aus und zeige die Meldung `FERTIG` an. Drücken Sie [R/S] zum Verlassen des Programms.

```
19 END
```


Indirekte Adressierung zum Ausführen von Subroutinen. Die nachstehende Routine verwendet XEQ IND, um in Subroutinen gespeicherte Daten (z.B. Telefonnummer) für einen eingegebenen Suchbegriff (z.B. Name) anzuzeigen.

Programm: / **Kommentar:**

```
00 { 137-Byte Prgm }
01 LBL "TEL#"
```

```
02 "NAME?"
03 AON
04 PROMPT
05 AOFF
06 ASTO ST X
07 XEQ IND ST X
08 PROMPT
```

Zeile 02 – 08: Eingabeaufforderung für Name (Alpha-String), dessen Telefonnummer gesucht werden soll (Zeile 02 – 05) und Speicherung des Strings im X-Register (Zeile 06). (Der String kann max. 6 Zeichen enthalten, da das X-Register maximal 6 Alphazeichen speichern kann. Ausführung der Subroutine, deren Label mit dem Alpha-String (Zeile 07) übereinstimmt, anschließend Unterbrechung des Programms (Zeile 08).

```
09 LBL "TOBIAS"
10 "07034-011482"
11 RTN
12 LBL "KATJA"
13 "07034-111383"
14 RTN
15 LBL "TAMMY"
16 "0711-062085"
17 RTN
18 LBL "OLIVER"
19 "0811-021054"
20 RTN
21 LBL "HP-BHG"
22 "06142-400-0"
23 RTN
```

Zeile 09 – 23: Speichere die Telefonnummern (eigentlich Alpha-Strings) im Alpha-Register.

```
24 END
```

Flags in Programmen

An früherer Stelle in diesem Kapitel haben Sie das Programm SSW erzeugt, welches, basierend auf dem Ausgang einer Abfrage, eine Verzweigung ausführt. SSW benutzt die Funktion X≤Y? zum Erzeugen der Verzweigung: *Ist* $S_2 \leq S_1$? Die Fortsetzung hängt von der Beantwortung dieser Abfrage ab – entweder wird der 2. Ergebnissatz berechnet oder das Programm wird abgeschlosssen.

Die Funktionssätze X?0 und X?Y erlauben Abfragen im Programm, die sich jedoch *nur* auf Zahlenwerte beziehen.* Allerdings können Programme auch bedingte Verzweigungen anstellen, die auf der Abfrage von *Flags* basieren. Diese Abfragen folgen der Regel "do-if-true", d.h. führe nächsten Schritt aus, wenn Abfrage mit "ja" bzw. "wahr" beantwortet wird; lautet die Antwort "nein" bzw. "falsch", wird die nächste Anweisung übersprungen. Da Flags eine eindeutige Bedeutung für den Rechner darstellen, können sie zur Erweiterung der Programmsteuerung verwendet werden. Die Benutzerflags 00 bis 35 und 81 bis 99 können gesetzt, gelöscht und abgefragt werden. Die Systemflags 36 bis 80 können nur abgefragt werden. (Für eine vollständige Auflistung der Flags und ihrer Bedeutung sollten Sie sich auf Anhang C im Benutzerhandbuch beziehen.)

Benutzerflags

Flags 00 bis 10 und 81 bis 99 dienen als *Benutzerflags*; Sie können diese setzen, löschen und abfragen. Diese Flags werden vom Rechner nicht intern verwendet und ihre Anwendungsweise bzw. Definition bleibt ausschließlich *Ihnen* überlassen.

* Ausnahmen bilden die Funktionen X=Y? und X≠Y?; diese können auch Alpha-Strings vergleichen.

Das Programm LIST auf Seite 176 bis 178 erzeugt die Matrix *ΣLIST*, wobei folgende Anweisungen verwendet werden:

```
31 LBL 02
32 1
33 ENTER
34 FC? 01
35 2
36 DIM "ΣLIST"
37 XEQ I
38 R↓
39 R↓
40 GTO 00
```

Vor der Ausführung von LIST muß Flag 01 *gesetzt* werden, falls *ΣLIST* eine einspaltige Matrix darstellen soll; wird eine zweispaltige Matrix gewünscht, muß Flag 01 *gelöscht* werden. (Denken Sie daran, daß der momentane Status eines Benutzerflags durch die andauernde Speicherfähigkeit des HP-42S beibehalten wird. Dies kann einen Einfluß auf andere Programme haben, welche den gleichen Flag benutzen.)

Steuerflags. Die *Steuerflags* 11 bis 35 besitzen eine spezielle Bedeutung und werden intern vom Rechner zur Darstellung verschiedener Betriebszustände benutzt. Ist z.B. Flag 21 *gesetzt*, während sich der Rechner im PROFF Modus befindet, dann werden VIEW und AVIEW Meldungen angezeigt und die Programmausführung wird unterbrochen. Befindet sich der Rechner dagegen im PRON Modus, werden VIEW und AVIEW Meldungen gedruckt und das Programm wird nicht angehalten. Viele der in diesem Handbuch vorgestellten Programme, die VIEW oder AVIEW verwenden, *setzen* auch Flag 21.

Systemflags

Die Systemflags 36 bis 80 werden vom Rechner zur Überwachung einer Reihe von Optionen und Bedingungen verwendet. Sie können diese Flags nicht beeinflussen, Sie können diese Flags jedoch abfragen.

Das nachstehende Programm MINMAX findet das kleinste oder größte Element einer Matrix im X-Register. In Zeile 23 wird der Status von Systemflag 77 (Anfang/Ende Umbruch) abgefragt, um festzustellen, ob das letzte Element der Matrix überprüft wurde.

MINMAX benutzt auch Benutzerflag 09 in Zeile 08, um festzustellen, ob das größte oder das kleinste Element gesucht werden soll. Vor dem Programmstart muß Flag 09 gesetzt werden, um das größte Element aufzufinden; ist Flag 09 gelöscht, wird das kleinste Element gesucht.

(Die kommentierte Programmliste finden Sie auf Seite 152 und 153.)

```
00 { 61-Byte Prgm }
01 LBL "MINMAX"

02 STO "MINMAX"
03 INDEX "MINMAX"

04 RCLEL
05 GTO 03

06 LBL 01
07 RCLEL
08 FS? 09
09 GTO 02
10 X≥Y?
11 GTO 04
12 GTO 03

13 LBL 02
14 X≤Y?
15 GTO 04

16 LBL 03
17 RCLIJ
18 RCL ST Z
19 ENTER

20 LBL 04
21 R↓
22 J+
23 FC? 77
24 GTO 01

25 END
```


Fehlerabfrage

Beim Versuch, eine unzulässige Operation während der Ausführung einer Funktion auszuführen, wird die Operation selbst nicht ausgeführt und es wird eine Fehlermeldung angezeigt. Beispielsweise bewirkt die Tastenfolge

1 [E] 260 ■ [x²]

die Anzeige der Meldung `Out of Range`, wobei der Wert 1×10^{260} im X-Register verbleibt.

Wird versucht, eine unzulässige Operation *in einem Programm* auszuführen, gibt der Rechner die entsprechende Meldung zurück und das Programm wird an der Stelle abgebrochen, die den Fehler verursacht hat. Betrachten Sie z.B. folgendes Programm:

```
00 { 26-Byte Prgm }
01 LBL "TRAP"
02 SF 21
03 INPUT "X"
04 X↑2
05 STO "Y"
06 VIEW "Y"
07 GTO "TRAP"
08 END
```

Bei der Ausführung von TRAP und Vorgabe des Wertes 1×10^{260} für X wird das Programm in Zeile 03 mit der Meldung `Out of Range` abgebrochen. Um einen neuen Wert für X vorzugeben, müssen Sie das Programm *erneut bei Zeile 01 starten*, indem Sie [XEQ] `TRAP` drücken. Bei einem kurzen Programm wie TRAP bereitet diese Art der Programmfortsetzung nach einem Fehler kein Problem. Wird jedoch ein komplexeres Programm ausgeführt (lange Rechenzeit oder wiederholte Dateneingabe), kann es sehr mühsam sein, nach jedem Fehler wieder bei Programmzeile 01 zu beginnen.

Sie können einem Programm erlauben, *nach dem Auftreten eines Fehlers fortzufahren*, indem Sie Flag 25 (Fehler ignorieren) setzen. Dies bewirkt:

- Während der Programmausführung wird *ein* Fehler ignoriert. Die fehlerverursachende Anweisung wird nicht ausgeführt und die Ausführung wird mit der nächsten Anweisung fortgesetzt.
- Der Fehler *löscht* Flag 25.

Beachten Sie diese Modifikation von TRAP.

```
00 { 58-Byte Prgm }
01 LBL "TRAP"

02 SF 21
03 SF 25
04 INPUT "X"
05 X↑2
06 FC?C 25
07 GTO 00
08 STO "Y"
09 VIEW "Y"
10 GTO "TRAP"

11 LBL 00
12 CF 21
13 BEEP
14 "Out of Range"
15 AVIEW
16 PSE
17 PSE
18 GTO "TRAP"

19 END
```

TRAP reagiert auf den Fehler nun wie folgt:

- Anzeige einer Fehlermeldung.
- Zurücksetzen von Flag 25 und Eingabeaufforderung für einen neuen *x*-Wert.

Diese Programmierungstechnik, als *Fehlerabfrage* bzw. *Fehlerbehandlung* bezeichnet, fügt zwar einige Programmschritte hinzu, ist aber sehr effizient, wenn das Auftreten eines Fehlerfalls wahrscheinlich ist.

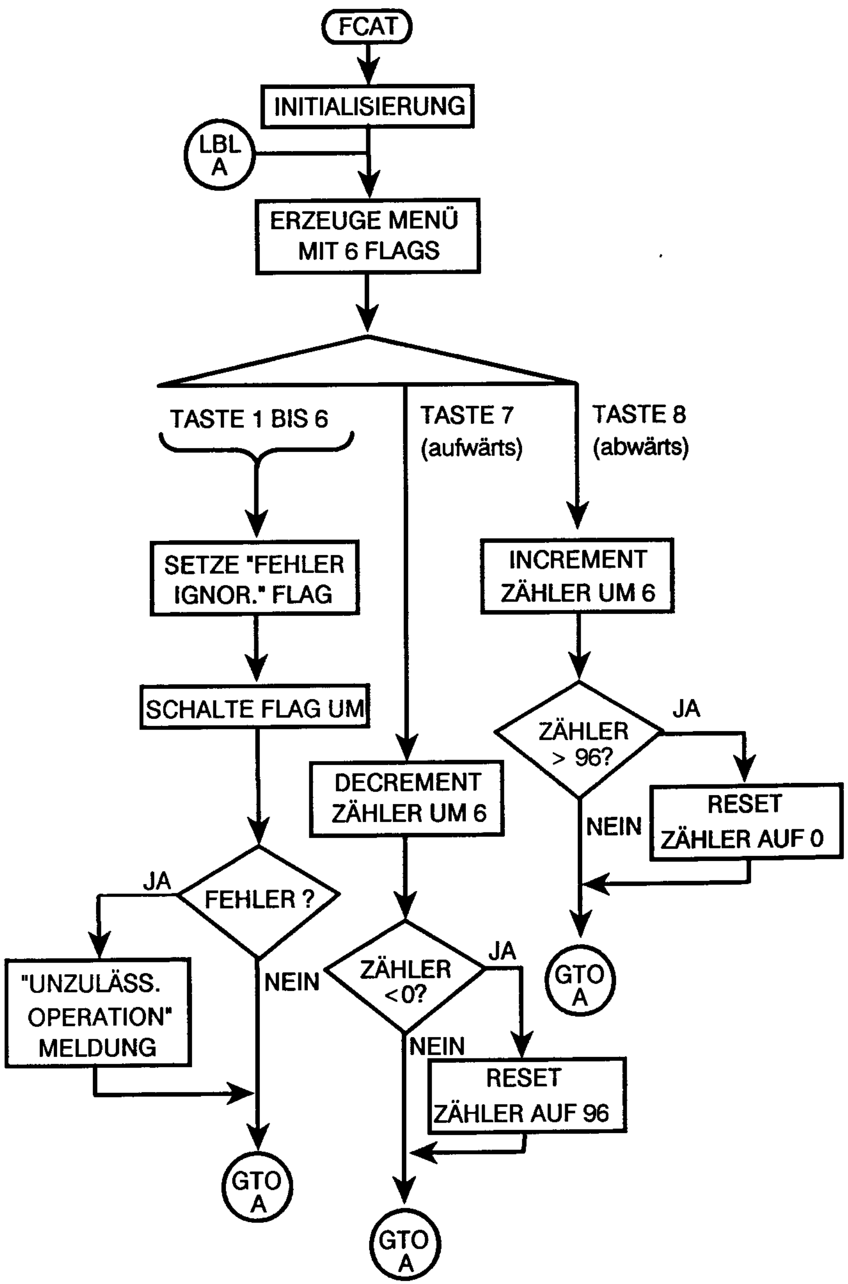
Ein zusammenfassendes Programm

Das folgende Programm FCAT zeigt den momentanen Status der Flags 00 bis 99 an. Die Flags werden in einem mehrzeiligen Menü mit jeweils 6 Feldern angezeigt. Jedes der Menüfelder enthält eine Flagnummer. Sie können die Benutzerflags 00 bis 35 (außer Flag 25) sowie 81 bis 99 setzen und löschen, indem Sie die entsprechende Menütaste drücken. Es wird das "■" Zeichen angezeigt, um zu kennzeichnen, daß der Flag momentan gesetzt ist. Wenn ein Systemflag gesetzt oder gelöscht werden soll, erzeugt FCAT ein Tonsignal und zeigt die Fehlermeldung `Unzuläss. Operation` an. Der vorangehende Satz mit 6 Flags läßt sich durch Drücken von Menütaste 7 ([▲]) anzeigen, der nächste Satz durch Drücken von Menütaste 8 ([▼]).

FCAT verwendet viele der Programmierungskonzepte, die in diesem Kapitel behandelt sind:

- Globale und lokale Labels.
- Eingabeaufforderung für Variablenwerte.
- Bedingte Verzweigungen, basierend auf:
  - Abfrage/Vergleich von Zahlen.
  - Abfrage von Flags.
- Subroutinen.
- Mehrzeilige Menüs.
- Bestimmte Schleifen.
- Indirekte Adressierung.
- Fehlerabfrage bzw. -Behandlung.

Nachstehend das Flußdiagramm für FCAT.



Hier die kommentierte Programmliste.

Programm:	**Kommentar:**
00 { 233-Byte Prgm } 01 LBL "FCAT"	
02 0,09606 03 STO 00	Zeile 02 – 03: Speichere die Schleifensteuerzahl in R_{00}.
04 LBL A 05 RCL 00 06 XEQ 00 07 KEY 1 GTO 01 08 XEQ 00 09 KEY 2 GTO 02 10 XEQ 00 11 KEY 3 GTO 03 12 XEQ 00 13 KEY 4 GTO 04 14 XEQ 00 15 KEY 5 GTO 05 16 XEQ 00 17 KEY 6 GTO 06	Zeile 4 – 17: Erzeuge die Menütasten 1 – 6. Das Label für jede Menütaste wird durch Aufruf von Subroutine 00 erzeugt. (Gehen Sie nun zu dieser über.)
18 KEY 7 GTO 07 19 KEY 8 GTO 08	Zeile 18 – 19: Weise Menütasten 7 und 8 GTO Anweisungen zu.
20 "FLAG CATALOG" 21 MENU 22 6 23 STO 01 24 PROMPT 25 GTO A	Zeile 20 – 25: Erzeuge den Alpha-String FLAG CATALOG (Zeile 20). Zeige das Menü an (Zeile 21). Initialisiere Register R_{01} mit 6 (Zeile 22 – 23). Zeige den Inhalt des Alpha-Registers an und unterbreche die Programmausführung zwecks Eingabe eines Zahlenwertes (Zeile 24).

```
26 LBL 00
27 CLA
28 99,1
29 X<>Y
30 X>Y?
31 RTN
32 AIP
33 FS? IND ST X
34 ⊢"■"
35 1
36 +
37 RTN
```

Subroutine 00, Zeile 26–37: Erzeuge den Alpha-String für jede Menütaste. Überprüfe zuerst, ob der momentane Wert im X-Register (Schleifenzähler) größer als 99 ist (Zeile 28–31). Falls ja, erzeuge *kein* Label für die Menütaste (99 ist die größte Flagnummer). Falls nein, hänge den ganzzahligen Teil des X-Registerwerts dem Inhalt im Alpha-Register an (Zeile 32). Frage den Status des Flags ab, dessen Nummer im Alpha-Register gespeichert ist. Ist dieser Flag *gesetzt*, hänge das "■" Zeichen dem Alpha-Registerinhalt hinzu (Zeile 33–34). Erhöhe den Inhalt im X-Register um 1 (Zeile 35–36).

```
38 LBL 01
39 DSE 01
40 LBL 02
41 DSE 01
42 LBL 03
43 DSE 01
44 LBL 04
45 DSE 01
46 LBL 05
47 DSE 01
48 LBL 06
49 DSE 01
50 LBL 14
51 RCL 01
52 RCL+ 00
```

Zeile 38–52 sorgen für das Löschen oder Setzen des Flags: Reduziere nacheinander R_{01} um 1 (Zeile 38–49). (Wird Menütaste 1 gedrückt, erhält man den Wert 0 in R_{01}, wenn R_{01} in das X-Register (Zeile 51) zurückgerufen wird. Wird Menütaste 6 gedrückt, erhält man als Wert in R_{01} 5, wenn R_{01} in das X-Register zurückgerufen wird.) Addiere den momentanen Wert in R_{00} (Zähler) zum momentanen Wert im X-Register (Zeile 52). (Nach Ausführung von Zeile 52 entspricht der Wert im X-Register der Nummer des zu löschenden oder zu setzenden Flags.)


```
53 SF 25
54 FC?C IND ST X
55 GTO 09
56 GTO A
```

Die Zeilen 53–56 erzeugen die Setzen/ Löschen-Funktion und Fehlerabfrage: Setze "Fehler ignorieren" Flag (Zeile 53). Abfrage, ob Flag (Nummer in X) gelöscht ist, und anschließendes Löschen des Flags. War der Flag bei der Abfrage *gelöscht, oder* führte der Versuch zum Löschen zu einer unzulässigen Operation, verzweige zu Label 09 (Zeile 55). War der Flag *gesetzt und* die Löschoperation führte zu keinem Fehler, kehre zur Menüfeld-Routine zurück und aktualisiere den Flagstatus (Zeile 56).

```
57 LBL 09
58 FC?C 25
59 GTO 10
60 SF IND ST X
61 GTO A
```

Zeile 57–61: Wenn die Verzweigung zu Label 09 durch eine unzulässige Operation verursacht wurde, gehe zu Label 10 (Zeile 57–59), ansonsten ist der Flag zu setzen und zur Menüfeld-Routine zurückzukehren, welche den Status aktualisiert (Zeile 60–61).

```
62 LBL 07
63 RCL 00
64 6
65 -
66 X<0?
67 96,09606
68 STO 00
69 GTO A
```

Zeile 62–69: *Reduziere* R_{00} um 6. (Wird also [▼] gedrückt, werden die obersten Menütasten umbenannt und zeigen eine Nummer an, die *um 6 kleiner* als im vorangehenden Menü ist. Wenn z.B. R_{00} 12 enthält und [▼] wird gedrückt, dann nimmt R_{00} den Wert 6 an und die Menütasten die Nummern 6–11.) Abfrage, ob R_{00} kleiner als Null ist. Falls ja, speichere 96 in R_{00} (Zeile 66–68). (Die Menütasten 1–4 nehmen die Nummern 96–99 an.)

```
70 LBL 08
71 ISG 00
72 GTO A
73 GTO "FCAT"
```

Zeile 70–73: *Erhöhe* R_{00} um 6 unter Anwendung der ISG Funktion. Denken Sie daran, daß die Zahl in R_{00} die Schleifensteuerzahl enthält; der ursprüngliche Wert war 0,09906. Wird [▲] gedrückt, werden die obersten Menütasten umnumeriert (mit einer Nummer, die um 6 *größer* als im vorangehenden Menü ist). Wird der Endwert der Schleifensteuerzahl (96) überschritten, verzweige zu FCAT, wobei der ursprüngliche Wert für die Schleifensteuerzahl zurückgespeichert wird; die Menütasten werden demnach mit 0–5 numeriert.

```
74 LBL 10
75 FS?C 21
76 GTO 11
77 XEQ 12
78 GTO A
79 LBL 11
80 XEQ 12
81 SF 21
82 GTO A
83 LBL 12
84 BEEP
85 "Unzuläss. "
86 ⊢"Operation"
87 AVIEW
88 PSE
89 RTN

90 END
```

Zeile 74–89: Führe BEEP Funktion aus und zeige Fehlermeldung `Unzuläss. Operation` an. Die Programmausführung springt danach zu Label A zurück. War Flag 21 gesetzt, lösche ihn vor der Anzeige der Fehlermeldung, und setze ihn anschließend wieder. (Die Programmausführung wird fortgesetzt, indem erneut das Flag-Menü angezeigt wird. *Außerdem* wird der Status von Flag 21 beibehalten.)

Beispiel: Das Flag-Katalog Programm. Verwenden Sie FCAT, um Flag 01 zu setzen. Fragen Sie den Status von Flag 38 ab; versuchen Sie, diesen Flag zu setzen oder zu löschen.

Starten Sie FCAT.

[XEQ] FCAT

```
FLAG CATALOG
 0 | 1 | 2 | 3 | 4 | 5
```

Setzen Sie Flag 01.

1

```
FLAG CATALOG
 0 | 1▪ | 2 | 3 | 4 | 5
```

Überprüfen Sie den Status von Flag 38.

[▼] [▼] [▼]
[▼] [▼] [▼]

```
FLAG CATALOG
 36 | 37▪ | 38 | 39 | 40▪ | 41
```

Flag 38 ist gelöscht. Versuchen Sie, ihn zu setzen.

38

```
FLAG CATALOG
 36 | 37▪ | 38 | 39 | 40▪ | 41
```

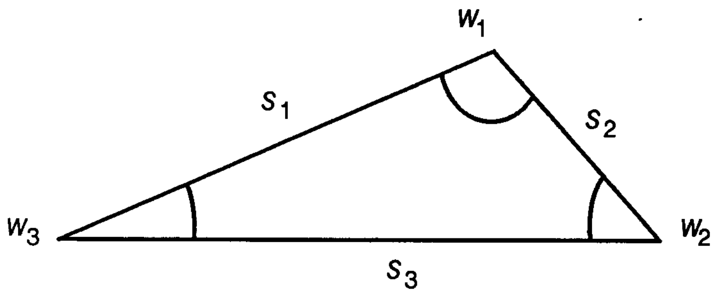
Der Rechner gibt ein Tonsignal aus und zeigt die Meldung `Unzuläss. Operation` an; danach wird der Status wie vor Beginn der Fehlersituation angenommen. Verlassen Sie FCAT.

[EXIT]

```
Y: 42,0961
X: 6,0000
```

Programm zum Lösen eines Dreiecks

Dieser Abschnitt enthält den vollständigen Gleichungssatz zum Lösen der Seiten bzw. Winkel eines Dreiecks. Außerdem sind die Anweisungen zum Eingeben des Programms 3ECK, eine kommentierte Liste von 3ECK und Hinweise zur Anwendung des Programms aufgeführt.



Im Programm verwendete Gleichungen. Nachstehende Gleichungen kommen im Programm zur Anwendung:

- Bedingung 1: S_1, S_2 und S_3 (3 Seiten) sind bekannt:

$$W_3 = 2 \text{ arc cos} \left[\frac{\sqrt{P(P - S_2)}}{(S_1 S_3)} \right] \text{ wobei } P = \frac{(S_1 + S_2 + S_3)}{2}$$

$$W_2 = 2 \text{ arc cos} \left[\frac{\sqrt{P(P - S_1)}}{(S_2 S_3)} \right]$$

$$W_1 = \text{arc cos} [-\cos (W_3 + W_2)]$$

- Bedingung 2: S_1, S_2 und W_2 (zwei Seiten und der angrenzende Winkel) sind bekannt:

$$W_3 = \text{arc sin} \left[\left(\frac{S_2}{S_1} \right) \sin W_2 \right]^*$$

$$W_1 = \text{arc cos} [-\cos (W_2 + W_3)]$$

Das Problem wurde auf die W_3, S_1, W_1 Konfiguration reduziert.

- Bedingung 3: W_3, S_1 und W_1 (2 Winkel und der eingeschlossene Winkel) sind bekannt:

$$W_2 = \text{arc cos} [-\cos (W_3 + W_1)]$$

$$S_2 = S_1 \left(\frac{\sin W_3}{\sin W_2} \right)$$

$$S_3 = S_1 \cos W_3 + S_2 \cos W_2$$

- Bedingung 4: S_1, W_1, and W_2 (1 Seite und die anschließenden Winkel) sind bekannt:

$$W_3 = \text{arc cos} [-\cos (W_1 + W_2)]$$

Das Problem wurde auf die W_3, S_1, W_1 Konfiguration reduziert.

- Bedingung 5: S_1, S_2 (2 Seiten und der eingeschlossene Winkel) sind bekannt:

$$S_3 = \sqrt{S_1^2 + S_2^2 - 2 S_1 S_2 \cos W_1}$$

Das Problem wurde auf die S_1, S_2, S_3 Konfiguration reduziert.

- Die Fläche für ein beliebiges Dreieck berechnet sich wie folgt:

$$AREA = \frac{1}{2} S_1 S_3 \sin W_3$$

* Es gibt zwei mögliche Lösungen, wenn S_2 größer als S_1 ist und W_3 ungleich 90° ist. Beide Ergebnissätze werden berechnet.

Um 3ECK einzutippen:

1. Erzeugen Sie die Variablen *S1, S2, S3, W1, W2, W3, P* und *AREA*, bevor Sie mit der Programmeingabe beginnen.
2. Erzeugen Sie die Labels RESULTS und SSWSUB, wenn Sie mit der Programmeingabe beginnen.

Nachstehend finden Sie eine kommentierte Programmliste von 3ECK.

Programm:	**Kommentar:**
00 { 573-Byte Prgm } 01 LBL "3ECK"	
02 SF 21	
03 "SSS" 04 KEY 1 XEQ A 05 "SSW" 06 KEY 2 XEQ B 07 "WSW" 08 KEY 3 XEQ C 09 "SWW" 10 KEY 4 XEQ D 11 "SWS" 12 KEY 5 XEQ E	Zeile 03 – 12: Lege die Zuweisungen für die Menütasten fest.
13 MENU 14 STOP 15 XEQ "RESULTS" 16 GTO "3ECK"	Zeile 13 – 16: Anzeige der Menütasten.
17 LBL A 18 INPUT "S1" 19 INPUT "S2" 20 INPUT "S3" 21 RCL "S1" 22 RCL+ "S2" 23 RCL+ "S3" 24 2 25 ÷ 26 STO "P" 27 X↑2 28 LASTX 29 RCL× "S2" 30 - 31 RCL "S1" 32 RCL× "S3" 33 ÷ 34 SQRT 35 ACOS 36 2 37 × 38 STO "W3" 39 SIN 40 RCL× "S1" 41 STO 00 42 RCL "P" 43 X↑2 44 LASTX 45 RCL× "S1" 46 - 47 RCL÷ "S2" 48 RCL÷ "S3" 49 SQRT 50 ACOS 51 2 52 × 53 STO "W2" 54 RCL+ "W3" 55 COS 56 +/- 57 ACOS 58 STO "W1" 59 RTN	Subroutine A, Zeile 17 – 59: Berechne die SSS Lösung.
60 LBL B 61 INPUT "S1" 62 INPUT "S2" 63 INPUT "W2" 64 SIN 65 RCL× "S2" 66 RCL÷ "S1"	Subroutine B, Zeile 60 – 100: Berechne die SSW Lösung.

```
67 ASIN
68 STO "W3"
69 SIN
70 RCL× "S1"
71 STO 00
72 XEQ "SSWSUB"
73 RCL "S1"
74 RCL "S2"
75 X≤Y?
76 RTN
77 XEQ "RESULTS"
78 RCL "W3"
79 COS
80 +/-
81 ACOS
82 STO "W3"
83 XEQ "SSWSUB"
84 RTN
85 LBL "SSWSUB"
86 RCL "W3"
87 RCL+ "W2"
88 COS
89 +/-
90 ACOS
91 STO "W1"
92 RCL "W2"
93 COS
94 RCL× "S2"
95 RCL "W3"
96 COS
97 RCL× "S1"
98 +
99 STO "S3"
100 RTN
```

Subroutine C, Zeile 101–126: Berechne die WSW Lösung.

```
101 LBL C
102 INPUT "W3"
103 INPUT "S1"
104 INPUT "W1"
105 RCL "W3"
106 RCL+ "W1"
```



```
107 COS
108 +/-
109 ACOS
110 STO "W2"
111 RCL "W3"
112 RCL "S1"
113 →REC
114 X<>Y
115 STO 00
116 RCL "W2"
117 1
118 →REC
119 R↓
120 ÷
121 STO "S2"
122 R↑
123 ×
124 +
125 STO "S3"
126 RTN
```

Subroutine D, Zeile 127–150: Berechne die SWW Lösung.

```
127 LBL D
128 INPUT "S1"
129 INPUT "W1"
130 INPUT "W2"
131 RCL+ "W1"
132 COS
133 +/-
134 ACOS
135 STO "W3"
136 RCL "S1"
137 →REC
138 X<>Y
139 STO 00
140 RCL "W2"
141 1
142 →REC
143 R↓
144 ÷
145 STO "S2"
146 R↑
```

```
147 ×
148 +
149 STO "S3"
150 RTN
```

Subroutine E, Zeile 151 – 194: Berechne die SWS Lösung.

```
151 LBL E
152 INPUT "S1"
153 INPUT "W1"
154 INPUT "S2"
155 RCL "W1"
156 X<>Y
157 →REC
158 RCL "S1"
159 -
160 →POL
161 STO "S3"
162 RCL+ "S1"
163 RCL+ "S2"
164 2
165 ÷
166 STO "P"
167 X↑2
168 LASTX
169 RCL× "S2"
170 -
171 RCL "S1"
172 RCL× "S3"
173 ÷
174 SQRT
175 ACOS
176 2
177 ×
178 STO "W3"
179 SIN
180 RCL× "S1"
181 STO 00
182 RCL "P"
183 X↑2
184 LASTX
185 RCL× "S1"
186 -
```



```
187 RCL÷ "S2"
188 RCL÷ "S3"
189 SQRT
190 ACOS
191 2
192 ×
193 STO "W2"
194 RTN
```

Subroutine RESULTS, Zeile 195 – 208: Berechne *AREA* und zeige die urspünglich bekannten Werte sowie die Ergebnisse an.

```
195 LBL "RESULTS"
196 RCL 00
197 RCL× "S3"
198 2
199 ÷
200 STO "AREA"
201 VIEW "S1"
202 VIEW "W1"
203 VIEW "S2"
204 VIEW "W2"
205 VIEW "S3"
206 VIEW "W3"
207 VIEW "AREA"
208 RTN

209 END
```

Anwendung von 3ECK:

1. Drücken Sie XEQ 3ECK.
2. Wählen Sie die gewünschte Lösung durch Drücken der entsprechenden Menütaste.
3. Geben Sie auf Anforderung den gefragten Eingabewert ein. Sie können eine beliebige Seite als S_1 bezeichnen. W_1 stellt den angrenzenden Winkel dar. Die Werte können im Uhrzeigersinn oder entgegengesetzt eingegeben werden; sie werden in der gleichen Reihenfolge angezeigt, wie sie eingegeben wurden.

2

Erweiterung von HP-41 Programmen

In Kapitel 11 des Benutzerhandbuchs ist ein Programm vorgestellt, welches ursprünglich für den HP-41 geschrieben wurde. Das Programm, als QUAD benannt, berechnet die (reellen) Lösungen von quadratischen Gleichungen. Zwei Programme in diesem Kapitel, Q2 und Q3, verwenden Leistungsmerkmale und Funktionen des HP-42S zur Erweiterung von QUAD. Ein drittes Programm, Q-KURZ, verwendet nur 11 Zeilen zur Lösung von quadratischen Gleichungen.

Verwenden von benannten Variablen

Im HP-42S können Daten in Datenspeicherregistern *oder* in benannten Variablen gespeichert und zurückgerufen werden. Werden in einem Programm Variablen zur Speicherung von Daten verwendet, so läßt sich i.d.R. das Programm einfacher schreiben und ist leichter zu verstehen.

In QUAD werden die Werte von den Koeffizienten a, b und c in Speicherregistern gespeichert und zurückgerufen. In Q2 werden diese Werte in *benannten Variablen* a, b und c gespeichert bzw. aus diesen zurückgerufen. (Q2 speichert außerdem die Werte der zwei Lösungen x_1 und x_2 in den Variablen *X1* und *X2* gespeichert. In QUAD werden diese Werte berechnet und angezeigt, jedoch nicht gespeichert.)

Anwendung von HP-42S Datenein- und Ausgabefunktionen

Eingabeaufforderung über INPUT

Die INPUT Funktion des HP-42S erlaubt in Programmen die Eingabe von Daten über eine *einzige* Programmzeile.

QUAD fordert zur Eingabe eines Wertes für *a* auf und speichert den Wert 2*a* in einem Speicherregister, wozu 3 Anweisungen verwendet werden.

```
02 "a=?"
03 PROMPT
06 STO 00
```

Q2 benutzt INPUT (und die benannte Variable *a*), um diese 3 Anweisungen mit einer zu ersetzen.

```
09 INPUT "a"
```

Anzeige von Daten über VIEW

Die VIEW Funktion des HP-42S ermöglicht in Programmen das Anzeigen von Daten über eine *einzige* Programmzeile.

QUAD zeigt den benannten Wert von x_1 über eine Reihe von 3 Anweisungen.

```
29 "X1= "
30 ARCL X
31 AVIEW
```

Q2 benutzt VIEW (und die benannte Variable *X1*), um diese 3 Anweisungen mit einer zu ersetzen.

```
33 VIEW "X1"
```


Operationen mit HP-42S Datentypen

Programme, die für den HP-41 Rechner geschrieben wurden, können nur 2 Datentypen bearbeiten: reelle Zahlen und Alpha-Strings. Programme für den HP-42S können jedoch auch komplexe Zahlen und Matrizen bearbeiten.

In QUAD können keine komplexe Lösungen berechnet werden. Stattdessen wird, falls $b^2 - 4ac$ kleiner als 0 ist, die Berechnung abgebrochen und die Meldung `LOESUNG KOMPLEX` angezeigt. In Q2 werden komplexe Lösungen berechnet, in Variablen gespeichert und angezeigt.

Verwenden der 2-zeiligen Anzeige

Programme können auch längere Meldungen in der 2-zeiligen Anzeige des HP-42S ausgeben. In Q2 wird die zweizeilige Meldung

```
Eingabe =0 unzulässig.
Drücken Sie R/S.
```

angezeigt, falls 0 als Eingabewert für eine der Variablen *a* oder *c* vorgegeben wurde.

Um Q2 einzugeben: Erzeugen Sie die Variablen *a*, *b*, *c*, *X1* und *X2*, bevor Sie mit der Programmeingabe beginnen.

Hier eine kommentierte Liste von Q2.

Programm:	**Kommentar:**

`00 { 132-Byte Prgm }`
`01 LBL 00`
`02 "Eingabe =0 u"`
`03 ├"nzulässig.␊Drü"`
`04 ├"cken Sie R/S."`
`05 PROMPT`
	Zeile 01–05: Zeige die "Eingabe =0" Fehlermeldung an.

```
06 LBL "Q2"
07 CPXES
08 SF 21
09 INPUT "a"
10 X=0?
11 GTO 00
12 INPUT "b"
13 INPUT "c"
14 X=0?
15 GTO 00
```

Zeile 06–15: Veranlasse das Programm, komplexe Lösungen zu berechnen; Aufforderung zur Eingabe von Werten für *a*, *b* und *c* sowie Test, ob 0 für *a* oder *c* vorgegeben wurde. (Flag 21 wird in Zeile 08 gesetzt, wodurch VIEW die Ergebnisse im PROFF Modus *anzeigt* bzw. *druckt*, falls PRON ausgeführt wurde.)

```
16 RCL "b"
17 +/-
18 ENTER
19 X↑2
20 4
21 RCL× "a"
22 RCL× "c"
23 -
24 SQRT
```

Zeile 16–24: Berechne

$\sqrt{b^2 - 4ac}$

```
25 RCL "b"
26 SIGN
27 ×
28 -
29 2
30 ÷
31 RCL÷ "a"
```

Zeile 25–31: Berechne entweder

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

oder

$$\frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

in Abhängigkeit vom Vorzeichen von *b*. Die Zeilen 25–27 stellen sicher, daß die Lösung mit dem größeren Absolutbetrag zuerst berechnet wird. Dadurch wird die Genauigkeit der Ergebnisse verbessert.

```
32 STO "X1"
33 VIEW "X1"
```

Zeile 32–33: Speichere den berechneten Wert in *X1* und zeige *X1* an.


```
34 RCL "c"
35 RCL÷ "a"
36 RCL÷ "X1"
37 STO "X2"
38 VIEW "X2"
```

Zeile 34–38: Berechne die zweite Lösung, speichere das Ergebnis in *X2* und zeige *X2* an.*

```
39 GTO "Q2"
```

Zeile 39: Rücksprung zum Label Q2.

```
40 END
```

Verwenden von Menüvariablen

Q2 verwendet die INPUT Funktion zur Eingabeaufforderung von Werten für die Variablen *a*, *b* und *c*. Q3 benutzt ein *Variablenmenü*, um Werte für diese Variablen einzugeben. In der nachstehenden kommentierten Liste sind die korrespondierenden Programmzeilen besonders hervorgehoben.

* Die quadratische Gleichung $ax^2 + bx + c = 0$ kann durch a dividiert werden (da a ungleich 0 sein muß), wodurch man $x^2 + \frac{bx}{a} + \frac{c}{a} = 0$ erhält. Diese Gleichung kann durch zwei Multiplikatoren in der Form $(x - X_1)(x - X_2)$ dargestellt werden, wobei X_1 und X_2 der Lösung der Gleichung entsprechen. Per Definition für den Ausgliederungsprozeß entspricht $(X_1)(X_2) = \frac{c}{a}$. Demzufolge entspricht $X_2 = \frac{c}{(aX_1)}$.

Um Q3 einzugeben: Erzeugen Sie die Variablen *a*, *b*, *c*, *X1* und *X2*, bevor Sie mit der Programmeingabe beginnen.

Programm:

```
00 { 143-Byte Prgm }
01 LBL 00
02 "Eingabe =0 u"
03 ⊢"nzulässig.␊Drü"
04 ⊢"cken Sie R/S."
05 PROMPT

06 LBL "Q3"
07 MVAR "a"
08 MVAR "b"
09 MVAR "c"
10 CPXES
11 SF 21
12 VARMENU "Q3"
13 STOP

14 RCL "a"
15 X=0?
16 GTO 00
17 RCL "c"
18 X=0?
19 GTO 00

20 RCL "b"
21 +/-
22 ENTER
23 X↑2
24 4
25 RCL× "a"
26 RCL× "c"
27 -
28 SQRT

29 RCL "b"
30 SIGN
31 ×
32 -
33 2
34 ÷
35 RCL÷ "a"

36 STO "X1"
37 VIEW "X1"

38 RCL "c"
39 RCL÷ "a"
40 RCL÷ "X1"
41 STO "X2"
42 VIEW "X2"

43 GTO "Q3"

44 END
```

Kommentar:

Zeile 06 – 13: Definiere die Menüvariablen *a*, *b* und *c* und erlaube dem Programm, komplexe Zahlen zu berechnen; setze Flag 21 und zeige das Variablenmenü an.

Programmzuweisung für CUSTOM Menüs

Beim Erzeugen des globalen Labels Q3 in Programmzeile 06 wurde dieses Label automatisch im HP-42S Programmkatalog aufgenommen. Sie können nun Q3 ausführen, indem Sie

[XEQ] Q3

drücken (was das Drücken von wenigstens 2 Tasten erfordert, je nachdem, wo sich das Label Q3 im Programmkatalog befindet).

Alternativ dazu können Sie Q3 dem CUSTOM Menü zuweisen, indem Sie

■[ASSIGN] PGM Q3

drücken und danach die gewünschte Menüzeile auswählen; anschließend ist die gewünschte Menütaste in dieser Zeile zu drücken. Das Programm läßt sich nun direkt vom CUSTOM Menü über einen einzelnen Tastendruck ausführen.

Beispiel: Ausführen eines erweiterten HP-41 Programms über CUSTOM Menü.

Teil 1. Führen Sie Q3 vom CUSTOM Menü aus, um die Lösung für die nachstehende Gleichung zu berechnen:

$$x^2 + 6x + 1 = 0 \,(a = 1, b = 6, c = 1)$$

Weisen Sie Q3 dem CUSTOM Menü zu, indem Sie die zuvor beschriebene Tastenfolge verwenden. Sollen die Ergebnisse gedruckt werden, dann ist PRON auszuführen. Starten Sie das Programm vom CUSTOM Menü aus.

(■ [PRINT] [▲] PON)
■ [CUSTOM] Q3

```
x: 0,0000
 A  B  C
```

Geben Sie die Werte für *a*, *b* und *c* ein. Berechnen Sie danach *X1*. (Falls die Ergebnisse gedruckt werden, entfällt deren Anzeige.)

1 A
6 B
1 C
[R/S]

```
X1=-5,8284
 A  B  C
```

X1 wird berechnet und angezeigt. Lösen Sie nun *X2*.

[R/S]

```
X2=-0,1716
 A  B  C
```

Kehren Sie zum Programmanfang zurück, um eine neue Berechnung ausführen zu können.

[R/S]

```
x: -0,1716
 A  B  C
```


Teil 2. Berechnen Sie die komplexe Lösung der Gleichung:

$$2x^2 + x + 3 = 0 \,(a = 2, b = 1, c = 3)$$

Spezifizieren Sie Rechtecksnotation. Geben Sie die Werte für *a*, *b* und *c* ein und berechnen Sie nun *X1*. (Falls die Ergebnisse gedruckt werden, entfällt deren Anzeige.)

■ [MODES] RECT
2 A
1 B
3 C
[R/S]

```
X1=-0,2500 -i1,1990
 A  B  C
```

X1 wird berechnet und angezeigt. Lösen Sie nun *X2*.

[R/S]

```
X2=-0,2500 i1,1990
 A  B  C
```

Verlassen Sie das Programm.

[EXIT]

```
Y: -0,2500 -i1,1990
x: -0,2500 i1,1990
```

Kurzversion des vorangehenden Programms. Zuletzt noch eine 11-zeilige Programmversion, die lediglich 26 Bytes beansprucht.

```
00 { 26-Byte Prgm }
01 LBL "Q-KURZ"

02 -0,5
03 ×
04 ENTER
05 ENTER
06 X↑2
07 RCL- ST T
08 SQRT
09 STO+ ST Z
10 -

11 END
```

Anwenden von Q-KURZ:

1. Ermöglichen Sie das Berechnen komplexer Ergebnisse durch Spezifizieren des entsprechenden Modus; als Koordinatenmodus ist Rechtecksnotation vorzugeben.
2. Tippen Sie den Wert für $\frac{c}{a}$ ein und drücken Sie [ENTER].
3. Tippen Sie den Wert für $\frac{b}{a}$ ein. Drücken Sie [XEQ] Q-KU.

3

Der Löser

Die in diesem Kapitel enthaltenen Informationen basieren auf dem Konzept, welches Ihnen in Kapitel 12 Ihres Benutzerhandbuchs vorgestellt wurde.

Es werden nachstehende Themen erläutert:

- Allgemeine Anwendungsweise des Lösers.
- Vorgabe von Anfangsnäherungen für den Löser.
- Emulation des Lösers.
- Anwendung des Lösers in Programmen.
- Einzelheiten zur Funktionsweise des Lösers.

Allgemeine Anwendungsweise des Lösers

Die allgemeine Vorgehensweise zur Anwendung des Lösers ist wie folgt:

1. Erzeugen Sie ein Programm, welches:
   a. MVAR zur Definition der Gleichungsvariable(n) verwendet.
   b. Die Gleichung so ausdrückt, daß die rechte Seite gleich 0 ist. (Beachten Sie, daß jede Variable in der Gleichung in das X-Register *zurückgerufen werden muß.*)
2. Wenden Sie den Löser für das Programm an:
   a. Drücken Sie ■ [SOLVER].
   b. Wählen Sie das Programm durch Drücken der korrespondierenden Menütaste.
   c. Geben Sie den Wert für jede bekannte Variable ein.
   d. Optional: Geben Sie eine oder zwei Anfangsnäherungen für die Unbekannte ein, indem Sie die Schätzwerte eintippen und die Menütaste für die Unbekannte drücken.

e. Lösen Sie die Unbekannte durch Drücken der entsprechenden Menütaste.

Beispiel: Einfache Anwendung des Lösers. Die Gleichung für den Zustand eines idealen Gases lautet:

$$PV = nRT$$

wobei:

P = Druck des Gases (in Atmosphären).
V = Volumen des Gases (in Liter).
n = Masse des Gases (in Mol).
R = allgemeine Gaskonstante (0,082057 l-atm/mol-K).
T = Temperatur des Gases (in Kelvin).

Teil 1. Erzeugen Sie ein Programm für den Löser, welches die Variablen festlegt und die Gleichung ausdrückt.

Setzen Sie zuerst die rechte Seite der Gleichung auf Null.

$$PV - nRT = 0$$

Schreiben Sie nun das entsprechende Programm.

Programm:	**Kommentar:**
00 { 42-Byte Prgm }	
01 LBL "GAS"	
02 MVAR "P"	Zeile 02–05: Definiere die Variablen.
03 MVAR "V"	
04 MVAR "n"	
05 MVAR "T"	
06 RCL "P"	Zeile 06–12: Drücke die Gleichung so aus, daß die rechte Seite gleich Null ist.
07 RCL× "V"	
08 RCL "n"	
09 RCL× "T"	
10 0,082057	
11 ×	
12 -	
13 END	

Teil 2. Verwenden Sie den Löser zum Lösen der nachstehenden Aufgabenstellung.

Berechnen Sie den Druck, welcher von 0,305 Mol Sauerstoff in 0,950 Liter bei einer Temperatur von 150 °C (423 K) ausgeübt wird. Es wird unterstellt, daß sich Sauerstoff wie ein ideales Gas verhält.

Wählen Sie die SOLVER Applikation.

■ [SOLVER] → Select Solve Program / GAS

Rufen Sie das zuvor erzeugte Programm auf.

GAS → x: 0,0000 / P V N T

Geben Sie die bekannten Variablenwerte ein.

,95 V
,305 N
423 T → T=423,0000 / P V N T

Berechnen Sie den Druck.

P → P=11,1438 / P V N T

Teil 3. Gleiches Volumen und Masse vorausgesetzt, wie hoch wäre die Temperatur bei 15 Atmosphären?

Da sich nur der Druck in der Aufgabenstellung geändert hat, müssen Sie nur dessen neuen Wert eingeben.

15 P → P=15,0000 / P V N T

Berechnen Sie nun die Temperatur.

T → T=569,3763 / P V N T

Verlassen Sie die SOLVER Applikation.

[EXIT] [EXIT] → y: 569,3763 / x: 569,3763

Vorgabe von Anfangsnäherungen für den Löser

Bei bestimmten Aufgabenstellungen ist es vorteilhaft, wenn Sie dem Löser zwei Anfangsnäherungen für die unbekannte Variable vorgeben. Sie erreichen dadurch eine kürzere Rechenzeit und Sie können den Löser auf einen bestimmten Lösungsbereich verweisen. Außerdem lassen sich dadurch weitere Lösungen (falls zutreffend) auffinden.

Verweisen des Lösers auf eine realistische Lösung

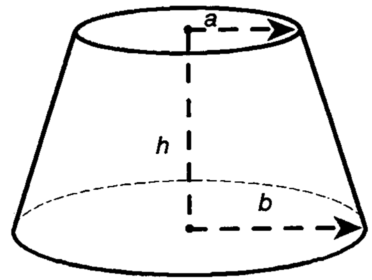
Es kommt oft vor, daß die Gleichung für den Löser ein System definiert, welches mehrere mathematisch korrekte Lösungen enthält; physikalisch betrachtet sind jedoch nur eine oder zwei von Bedeutung. In diesen Fällen kann es erforderlich sein, daß Sie den Löser auf den Bereich einer realistischen Lösung hinweisen bzw. einschränken, indem Sie geeignete Anfangsnäherungen (Schätzwerte) vorgeben.

Beispiel: Verweisen des Lösers auf eine realistische Lösung. Das Volumen eines geraden Kegelstumpfes läßt sich wie folgt berechnen:

$$V = \frac{1}{3}\pi h\,(a^2 + ab + b^2)$$

wobei:

V = Volumen des Stumpfes.
h = Höhe des Stumpfes.
a = Radius der Stumpfdecke.
b = Radius der Stumpfgrundfläche.



Teil 1. Schreiben Sie ein Programm, welches die Variablen definiert und die Gleichung in der Weise ausdrückt, daß die rechte Seite gleich Null ist.

```
00 { 46-Byte Prgm }
01 LBL "KEGEL"

02 MVAR "V"
03 MVAR "h"
04 MVAR "a"
05 MVAR "b"

06 RCL "a"
07 X↑2
08 LASTX
09 RCL× "b"
10 +
11 RCL "b"
12 X↑2
13 +
14 RCL× "h"
15 PI
16 ×
17 3
18 ÷
19 RCL- "V"

20 END
```

Für das vorliegende Beispiel soll angenommen werden, daß Sie die Variable a bereits erzeugt und in einem vorherigen Programm verwendet haben. Nehmen Sie an, es wäre $-3{,}7765$ momentan in a gespeichert. (Fahren Sie fort, indem Sie den Wert durch Drücken von 3,7765 [+/-] [STO] A in a speichern.)

Teil 2. Um den Radius a eines Kegelstumpfs mit dem Volumen V = 119,381 m^3, der Höhe h = 6 m und einem Grundradius b = 3 m zu berechnen, ist der Löser über das zugehörige Programm zu verwenden.

Wählen Sie die SOLVER Applikation und danach das Programm KEGEL.

■[SOLVER] KEGE

```
x: -3,7765
 V | H | A | B | | 
```

Geben Sie die Werte der bekannten Variablen ein.

119,381 V
6 H
3 B

```
b=3,0000
 V | H | A | B | | 
```

Berechnen Sie a.

A

```
a=-5,0000
 V | H | A | B | | 
```

Der Löser verwendet den momentanen Inhalt von Variable a ($-3{,}7765$) als Anfangsnäherung und ermittelt als Lösung $a = -5$ m. Das Ergebnis ist *mathematisch korrekt*. Allerdings hat ein negativer Radius physikalisch keine Bedeutung. Versuchen Sie es mit den Anfangsnäherungen 0 und 5.

0 A
5 A
A

```
a=2,0000
 V | H | A | B | | 
```

Der Wert 2,0000 für den Radius a ist mathematisch richtig und von physikalischer Bedeutung.

Verlassen Sie den Löser.

[EXIT] [EXIT]

```
y: 2,0000
x: 2,0000
```


Auffinden mehrerer Lösungen

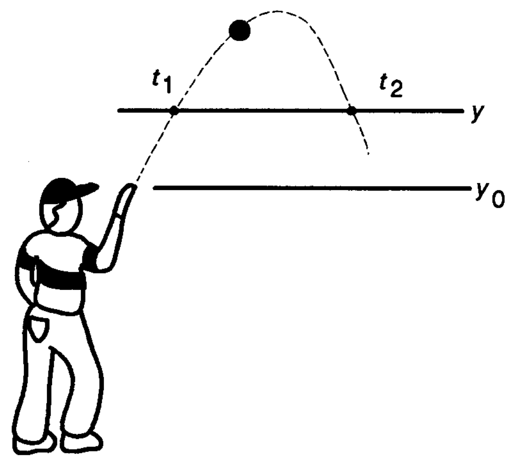
Die Gleichung für den Bewegungsablauf eines Objekts, welches der Schwerebeschleunigung ausgesetzt ist, lautet:

$$y = y_0 + v_0 t + \frac{1}{2} g t^2$$

wobei:

y = zurückgelegte Strecke
y_0 = Anfangsposition
v_0 = Anfangsgeschwindigkeit
g = Schwerebeschleunigung ($-9{,}8$ m/s^2).
t = verbrauchte Zeit

Ihr Benutzerhandbuch enthält unter "Weitere Löser-Beispiele" in Kapitel 12 mehrere Aufgabenstellungen, in welchen ein Objekt aus einer Anfangsposition *fallen gelassen* wird; v_0 entspricht hierbei 0 und die Richtung der zurückgelegten Strecke weist senkrecht nach unten. Ein Objekt, welches *nach oben* geworfen wird, erreicht eine gewisse Höhe y bei *zwei* unterschiedlichen Zeiten – einmal aufwärts gerichtet und zum anderen abwärts gerichtet.



Um beide Zeiten, t_1 und t_2 zu finden, müssen Sie den Löser zweimal anwenden, wobei wenigstens einmal eine Anfangsnäherung vorzugeben ist, um den Löser auf die zweite Lösung hinzuführen.

Beispiel: Anwenden des Lösers zum Auffinden zweier realistischer Lösungen. Ein Ball wird mit einer Anfangsgeschwindigkeit von v_0 = 15 m/s aus einer Anfangshöhe von y_0 = 2 m nach oben geworfen. Verwenden Sie den Löser zur Berechnung der zwei Zeiten t_1 und t_2, bei welchen der Ball die Höhe y = 5 m erreicht hat.

Teil 1. Erzeugen Sie ein Löser-Programm, welches die Variablen definiert und die Gleichung so ausdrückt, daß die rechte Seite gleich Null ist.

```
00 { 51-Byte Prgm }
01 LBL "WURF"

02 MVAR "y"
03 MVAR "y0"
04 MVAR "v0"
05 MVARA "t"

06 RCL "y0"
07 RCL "v0"
08 RCL× "t"
09 RCL "t"
10 X↑2
11 -9.8
12 ×
13 2
14 ÷
15 +
16 +
17 RCL- "y"

18 END
```


Teil 2. Verwenden Sie den Löser zur Berechnung der ersten Zeit t_1. Da Sie annehmen können, daß diese Zeit nahe Null ist, können Sie als Anfangsnäherungen 0 und 1 vorgeben.

Wählen Sie die SOLVER Applikation und danach das Programm WURF.

■ [SOLVER] [WURF]

```
x: 0,0000
 Y  Y0  V0  T
```

Geben Sie die Werte für die bekannten Variablen ein.

5 [Y]
2 [Y0]
15 [V0]

```
v0=15,0000
 Y  Y0  V0  T
```

Berechnen Sie t_1 unter Verwendung der Anfangsnäherungen 0 und 1.

0 [T]
1 [T]
[T]

```
t=0,2151
 Y  Y0  V0  T
```

Der Löser findet als Ergebnis für t_1 den Wert 0,2151 Sekunden. Berechnen Sie nun die zweite Zeit t_2 durch Vorgabe zweier Anfangsnäherungen, von welchen Sie annehmen, daß Sie die 2. Lösung eingrenzen; die Werte 1 und 20 erscheinen hierfür geeignet. (Sie müssen die anderen Variablenwerte nicht erneut eingeben, da sie sich nicht geändert haben.)

1 [T]
20 [T]
[T]

```
t=2,8461
 Y  Y0  V0  T
```

Der Löser berechnet als Ergebnis für t_2 = 2,8461 Sekunden.

Verlassen Sie den Löser.

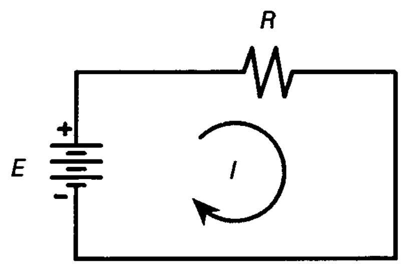
[EXIT] [EXIT]

```
y: 2,8461
x: 2,8461
```

Emulation des Lösers

Bei bestimmten Arten von Funktionen kann der Löser keine Lösung auffinden (z.B. läßt sich der Löser nicht für komplexe Zahlen anwenden). Sie können für solche Funktionen jedoch ein Programm schreiben, welches *explizit* zu einer Lösung führt und welches sich während der Ausführung *wie der Löser verhält*.

Betrachten Sie als Beispiel nachstehenden einfachen Schaltkreis.



Das Ohmsche Gesetz definiert die Beziehung zwischen der Spannung E, dem Widerstand R und dem Stom I in einem elektronischen Schaltkreis. Die Gleichung lautet:

$$E = IR$$

Da die Gleichung keine Terme für komplexe Zahlen enthält, läßt sich der Löser zum Auffinden einer beliebigen Gleichungsvariablen einsetzen.

Beispiel: Verwenden des Lösers für einfachen ohmschen Schaltkreis. Berechnen Sie mit Hilfe des Lösers den Widerstand R in einem einfachen ohmschen Schaltkreis, wenn die Spannung $E = 10$ V und die Stromstärke $I = 5$ A beträgt.

Erzeugen Sie zuerst ein Löser-Programm, welches die Variablen definiert und die Gleichung so ausdrückt, daß die rechte Seite gleich 0 ist.

```
00 { 25-Byte Prgm }
01 LBL "OHM"

02 MVAR "E"
03 MVAR "I"
04 MVAR "R"

05 RCL "I"
06 RCL× "R"
07 RCL- "E"

08 END
```

Wählen Sie zuerst die SOLVER Applikation und anschließend das Programm OHM.

■ [SOLVER] OHM

```
x: 0,0000
E I R
```

Geben Sie die bekannten Variablenwerte für E und I ein und berechnen Sie danach R.

10 E
5 I
R

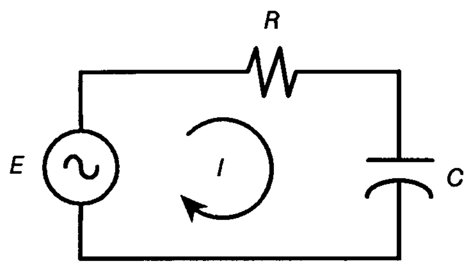
```
R=2,0000
E I R
```

Verlassen Sie den Löser.

[EXIT] [EXIT]

```
y: 2,0000
x: 2,0000
```

Betrachten Sie nun den nachstehenden Schaltkreis.



Die Anwendung des Ohmschen Gesetzes auf diesen Schaltkreis führt zu folgender Gleichung:

$$E = IZ$$

wobei:

E = Klemmenspannung
I = Stromstärke
Z = Impedanz

Die *Impedanz Z* wird durch die komplexe Zahl (in Rechtecksnotation)

$$R - i\left(\frac{1}{\omega C}\right)$$

dargestellt, wobei:

R = ohmscher Widerstand des Schaltkreises
ω = Frequenz des Schaltkreises (in Radiant/Sekunde).
C = Kapazität des Schaltkreises

Da die Spannung, Stromstärke und Impedanz komplexe Größen darstellen, ist die Verwendung des Lösers hier nicht möglich. Der Rechner kann jedoch *arithmetische* Operationen mit komplexen Zahlen durchführen. (Beziehen Sie sich auf Kapitel 6 im Benutzerhandbuch zur Erläuterung von Arithmetik mit komplexen Zahlen im HP-42S.) Das folgende Programm EIZ löst explizit (algebraisch) die Werte für die komplexen Größen E, I und Z und benutzt ein *Variablenmenü* zur Simulation der Anwendungsweise des Lösers.

Nachstehend ein kommentierter Ausdruck des Programms.

Programm:	**Kommentar:**
`00 { 96-Byte Prgm }` `01 LBL "EIZ"`	

`02 MVAR "E∡"`
`03 MVAR "I∡"`
`04 MVAR "Z∡"`
`05 VARMENU "EIZ"`
	Zeile 02 – 05: Definiere die Variablen E, I und Z und erzeuge das Variablenmenü.

`06 POLAR`
`07 CPXRES`
`08 CLA`
`09 STOP`
`10 ALENG`
`11 X=0?`
`12 GTO "EIZ"`
	Zeile 06 – 12: Spezifizieren von Polarnotation für Koordinatendarstellung und Berechnung von komplexen Ergebnissen. Unterbreche die Programmausführung für Dateneingabe. Falls keine zu lösende Variable eingegeben wurde, Rückkehr zum Programmanfang.

`13 ASTO ST X`
`14 XEQ IND ST X`
`15 STO IND ST Y`
`16 VIEW IND ST Y`
`17 GTO "EIZ"`
	Zeile 13 – 17: Rufe den momentanen Alpha-String in das X-Register zurück und führe korrespondierende Subroutine aus. (Momentaner Alpha-String ist die Variable, für welche kein Wert eingegeben wurde.) Speichere berechnetes Ergebnis im Y-Register und zeige Ergebnis an. Anschließend Rückkehr zum Programmanfang.

`18 LBL "E∡"`
`19 RCL "I∡"`
`20 RCL× "Z∡"`
`21 RTN`
	Subroutine E∡, Zeile 18 – 21: Berechne E∡ in Abhängigkeit von I∡ und R∡.

`22 LBL "I∡"`
`23 RCL "E∡"`
`24 RCL÷ "Z∡"`
`25 RTN`
	Subroutine I∡, Zeile 22 – 25: Berechnet I∡ in Abhängigkeit von E∡ und Z∡.

```
26 LBL "Z∡"
27 RCL "E∡"
28 RCL÷ "I∡"
29 RTN

30 END
```

Subroutine Z∡, Zeile 26–29: Berechne Z∡ in Abhängigkeit von E∡ und I∡.

(Zeile 06 spezifiziert Polarnotation für die Interpretation von Koordinaten. Meßgeräte zeigen gewöhnlich Spannung, Stromstärke und Impedanz in Polarnotation, d.h. als Betrag und Phasenverschiebung, an.)

Beispiel: Berechnen komplexer Größen in einem RC-Schaltkreis. Eine Netzteil mit einer Ausgangsspannung von 10 V und einer Phasenverschiebung von 0° versorgt einen RC-Schaltkreis bei einer Frequenz von 40 rad/s. Es wird eine Stromstärke von 0,37 A mit einer Phasenverschiebung von 68° gemessen. Wie groß ist die Kapazität und der Widerstand des Schaltkreises?

Aufruf des Programms EIZ.

[XEQ] EIZ

```
x: 0,0000
 E∡ I∡ Z∡
```

Geben Sie den Wert für die Ausgangsspannung des Netzteils ein.

10 [ENTER] 0 ■[COMPLEX] E∡

```
E∡=10,0000 ∡0,0000
 E∡ I∡ Z∡
```

Geben Sie den Wert für die Stromstärke ein.

,37 [ENTER] 68 ■[COMPLEX] I∡

```
I∡=0,3700 ∡68,0000
 E∡ I∡ Z∡
```

Berechnen Sie die Impedanz.

Z∡

```
Z∡=27,0270 ∡-68,0000
 E∡ I∡ Z∡
```


Die Impedanz des Schaltkreises (in Polarnotation) beträgt 27 Ω bei einer Phasenverschiebung von −68°. Stellen Sie die Impedanz in Rechtecksnotation dar, um den Widerstand und die Kapazität zu berechnen. (Denken Sie daran, daß *R* den Realteil darstellt, während *C* ein Faktor des imaginären Terms der Impedanz *Z* in *Rechtecksnotation* darstellt.)

■[MODES] RECT

```
x: 10,1245 -i25,0590
 E∡ I∡ Z∡
```

Der ohmsche Widerstand im Schaltkreis beträgt 10 Ω. Berechnen Sie nun die Kapazität.

■[COMPLEX]
[+/-] 40 [×]
■[TOP.FCN] 1/x

```
x: 0,0010
 E∡ I∡ Z∡
```

Die Kapazität des Schaltkreises beträgt 0,001 F.

Wenn nun als Impedanz (bei ursprünglicher Klemmenspannung) der Wert 20 Ω bei einer Phasenverschiebung von −45° gemessen wird, wie groß ist dann die Stromstärke?

Spezifizieren Sie wieder Polarnotation. Geben Sie danach den neuen Impedanzwert ein und berechnen Sie die Stromstärke.

■[MODES] POLAR
20 [ENTER] 45 [+/-]
■[COMPLEX] Z∡
I∡

```
I∡=0,5000 ∡45,0000
 E∡ I∡ Z∡
```

Die Stromstärke beträgt 0,5 A bei einer Phasenverschiebung von 45°.

Verlassen Sie EIZ.

[EXIT]

```
Y: "I∡"
x: 0,5000 ∡45,0000
```

Anwendung des Lösers in Programmen

Verwenden des Lösers und explizite Lösungen in einem Programm

Der Löser verwendet einen iterativen Algorithmus zum Auffinden der Lösungen für die Variablen einer Gleichung. Sie *müssen* ein iteratives Verfahren zum Lösen einer Variablen verwenden, wenn sich die Variable in der Gleichung nicht isolieren läßt (nicht ausschließlich in Abhängigkeit der anderen Variablen ausdrückbar). Läßt sich jedoch die Variable durch eine arithmetische Operation isolieren, so ist eine explizite Lösung dieser Variablen *immer schneller* als eine iterative Lösung über den Lösers.

Einige Funktionen können eine Variable enthalten, die über ein iteratives Verfahren zu lösen ist, sowie Variablen, deren Werte explizit berechnet werden können. In Kapitel 12 des Benutzerhandbuchs ist eine Gleichung für den Löser vorgestellt, die zur Bearbeitung von finanzmathematischen Problemstellungen (TVM) dient. Die angesprochene Gleichung lautet:

$$0 = BARW + (1 + ip)\,RATE \left[\frac{1 - (1+i)^{-N}}{i}\right] + ENDW\,(1+i)^{-N}$$

wobei:

- N = Anzahl der Verzinsungsperioden oder Zahlungen (RATE).
- i = periodischer Zinssatz in Dezimalform.
- $BARW$ = Barwert (kann sich auf eine Reihe zukünftiger Zahlungen oder auf eine ursprüngliche Investitionssumme beziehen). BARW tritt immer zum Beginn der ersten Periode auf.
- $RATE$ = die Höhe der periodischen Zahlung.
- $ENDW$ = Endwert (kann sich auf die Höhe des letzten Cashflows oder auf eine Reihe früherer Zahlungen unter Berücksichtigung des Verzinsungseffekts beziehen). ENDW tritt immer am Ende der n-ten Periode auf.
- p = Zahlweise. Wenn $p = 1$, dann werden die Zahlungen am *Anfang* jeder Periode geleistet. Ist $p = 0$, so treten die Zahlungen am *Ende* jeder Periode auf.

Das Beispiel, welches im Benutzerhandbuch aufgeführt ist, enthält das Programm ANNU, das jede Variable der TVM Gleichung definiert sowie die Gleichung selbst ausdrückt. Zum Lösen jeder in der Gleichung enthaltenen Variable war der Löser zu benutzen. Beachten Sie jedoch, daß jede Variable isoliert werden kann. So läßt sich BARW z.B. als

$$BARW = -(1 + ip)\,RATE \left[\frac{1 - (1+i)^{-N}}{i}\right] - ENDW\,(1+i)^{-N}$$

ausdrücken. Nur i kann nicht isoliert werden; Sie müssen den Löser nur dann benutzen, wenn Sie den Wert von i berechnen möchten.

Das folgende Programm ANN2 berechnet explizit die Lösungen für *BARW*, *N*, *ENDW* und *RATE* und ruft den Löser zur Berechnung von i auf. Das Programm verwendet ein programmierbares Menü und Flag 22 (numerische Dateneingabe), um das externe Erscheinen der SOLVER Applikation zu simulieren.

Um ANN2 einzutippen: Erzeugen Sie die Variablen *#R/J*, *p*, *STRG*, *N*, *ENDW*, *MODE*, *RATE*, *i*, *I%JR* und *BARW*.

Hier nun ein kommentierter Ausdruck des Programms.

Programm: **Kommentar:**

```
00 { 565-Byte Prgm }
01 LBL "ANN2"

02 REALRES
03 CF 21
04 12
05 SF 25
06 RCL "#R/J"
07 XEQ 21
08 SF 25
09 RCL "p"
10 CF 25
11 1
12 X≠Y?
13 0
14 STO "p"
15 XEQ 20
```

Zeile 02 – 15: Stelle sicher, daß Ergebnisse reellwertig sind. Zeige AVIEW Meldungen an und setze Programm fort. Rufe Subroutine 21 auf, um als Voreinstellung 12 für Anzahl der Zahlungen (#R/J) sowie Endmodus für Zahlweise einzustellen. Rufe Subroutine 20 auf, um Zahlungen pro Jahr und die Zahlweise anzuzeigen.

```
16 LBL 99
17 CLMENU
18 "N"
19 KEY 1 XEQ 01
20 "I%JR"
21 KEY 2 XEQ 02
22 "BARW"
23 KEY 3 XEQ 03
24 "RATE"
25 KEY 4 XEQ 04
26 "ENDW"
27 KEY 5 XEQ 05
28 "MODE"
29 KEY 6 GTO 06
30 MENU
31 STOP
32 ASTO "STRG"
33 STO IND "STRG"
34 VIEW IND "STRG"
35 GTO 99
```

Zeile 16–35: Erzeuge das Hauptmenü, zeige es an und warte auf Dateneingabe (Zeile 17–31). Zeige den Wert der eingegebenen oder berechneten Variable an (Zeile 32–34).

```
36 LBL 20
37 CLA
38 RCL "#R/J"
39 AIP
40 ⊢" #R/J"
41 RCL "p"
42 X=0?
43 ⊢"  END-MODUS"
44 X≠0?
45 ⊢" BEGIN-MODUS"
46 AVIEW
47 CLMENU
48 RTN
```

Subroutine 20, Zeile 36–48: Erzeugen und Anzeigen der Meldung für Zahlungen/Jahr und Zahlweise.


```
49 LBL 06
50 XEQ 20
51 "#R/J"
52 KEY 1 XEQ 21
53 "BEG"
54 KEY 2 XEQ 22
55 "END"
56 KEY 3 XEQ 23
57 "ANNU"
58 KEY 4 GTO "ANN2"
59 MENU
60 RCL "#R/J"
61 STOP
62 GTO 06
```

Zeile 49–62: Erzeugen und Anzeigen des Menüs für Zahlungen/Jahr und Zahlweise.

```
63 LBL 21
64 ABS
65 IP
66 1000
67 X<>Y
68 X≥Y?
69 12
70 X=0?
71 12
72 STO "#R/J"
73 RTN
```

Subroutine 21, Zeile 63–73: Überprüfe, ob der Wert für #R/J zulässig ist. Falls nicht, substituiere 12 Zahlungen/Jahr.

```
74 LBL 22
75 1
76 STO "p"
77 RTN
```

Subroutine 22, Zeile 74–77: Spezifiziere Beginn-Modus als Zahlweise durch Vorgabe von 1 für *p*.

```
78 LBL 23
79 0
80 STO "p"
81 RTN
```

Subroutine 23, Zeile 78–81: Spezifiziere End-Modus als Zahlweise durch Vorgabe von 0 für *p*.

Subroutine 01, Zeile 82 – 107: Wenn für *N* eine numerische Eingabe erfolgte, Rückkehr zum Hauptmenü und Anzeige des Wertes für *N*. Wenn nicht, berechne *N* in Abhängigkeit von den anderen Variablen. Falls $i = 0$, Sprung zu Label 00, um *N* (Zeile 93 – 95) zu berechnen.

```
82 LBL 01
83 "N"
84 FS?C 22
85 RTN
86 1
87 STO "N"
88 XEQ 10
89 RCL "ENDW"
90 RCL÷ "MODE"
91 +/-
92 RCL "RATE"
93 RCL "i"
94 X=0?
95 GTO 00
96 ÷
97 +
98 LASTX
99 RCL "BARW"
100 RCL÷ "MODE"
101 +
102 ÷
103 LN
104 RCL "i"
105 LN1+X
106 ÷
107 RTN
```

Subroutine 00, Zeile 108 – 113: Berechnet *N*, falls $i = 0$.

```
108 LBL 00
109 RCL "BARW"
110 RCL+ "ENDW"
111 RCL÷ "RATE"
112 +/-
113 RTN
```


Subroutine 02, Zeile 114 – 123: Verwende den Löser zur Berechnung von *I%JR*. Spezifiziere die Löser-Subroutine "i". Vorgabe der Anfangsnäherungen 0 und 20 für *I%JR*.

```
114 LBL 02
115 "I%JR"
116 FS?C 22
117 RTN
118 PGMSLV "i"
119 0
120 STO "I%JR"
121 20
122 SOLVE "I%JR"
123 RTN
```

Subroutine "i", Zeile 124 – 131: Drücke die TVM Gleichung für den Löser aus.

```
124 LBL "i"
125 XEQ 10
126 RCL× "RATE"
127 X<>Y
128 RCL× "ENDW"
129 +
130 RCL+ "BARW"
131 RTN
```

Subroutine 03, Zeile 132 – 142: Wenn für *BARW* eine numerische Eingabe erfolgte, Rückkehr zum Hauptmenü und Anzeige des Wertes für *BARW*. Falls nicht, berechne *BARW* in Abhängigkeit von den anderen Variablen.

```
132 LBL 03
133 "BARW"
134 FS?C 22
135 RTN
136 XEQ 10
137 RCL× "RATE"
138 X<>Y
139 RCL× "ENDW"
140 +
141 +/-
142 RTN
```

Subroutine 04, Zeile 143 – 154: Wenn für *RATE* eine numerische Eingabe erfolgte, Rückkehr zum Hauptmenü und Anzeige des Wertes für *RATE*. Falls nicht, berechne *RATE* in Abhängigkeit von den anderen Variablen.

```
143 LBL 04
144 "RATE"
145 FS?C 22
146 RTN
147 XEQ 10
148 X<>Y
149 RCL× "ENDW"
150 RCL+ "BARW"
```

```
151 X<>Y
152 ÷
153 +/-
154 RTN
```

```
155 LBL 05
156 "ENDW"
157 FS?C 22
158 RTN
159 XEQ 10
160 RCL× "RATE"
161 RCL+ "BARW"
162 X<>Y
163 ÷
164 +/-
165 RTN
```

Subroutine 05, Zeile 155–165: Wenn für *ENDW* eine numerische Eingabe erfolgte, Rückkehr zum Hauptmenü und Anzeige des Wertes für *ENDW*. Falls nicht, berechne *ENDW* in Abhängigkeit von den anderen Variablen.

```
166 LBL 10
167 RCL "I%JR"
168 RCL÷ "#R/J"
169 100
170 ÷
171 STO "i"
172 RCL× "p"
173 1
174 +
175 STO "MODE"
176 1
177 ENTER
178 RCL+ "i"
179 RCL "N"
180 +/-
181 Y↑X
182 STO ST Z
183 -
184 RCL× "MODE"
185 SF 25
186 RCL÷ "i"
187 FS?C 25
188 RTN
```

Subroutine 10, Zeile 166–188: Berechne Terme der TVM Gleichung, basierend auf dem Wert von *I%JR*. Berechne *i*, die Dezimalform des periodischen Zinssatzes (Zeile 167–171). Berechne *MODE* $(1 + ip)$ (Zeile 172–175). Berechne den *ENDW* Koeffizient $(1 + i)^{-N}$ (Zeile 176–182). Berechne den *RATE* Koeffizient. Falls $i = 0$, Sprung zu Zeile 189 (Zeile 183–188).


```
189 1
190 RCL "N"
191 END
```

Zeile 189–191: Falls $i = 0$, dann ist der *ENDW* Koeffizient gleich 1 und der *RATE* Koeffizient gleich *N*.

Anwenden von ANN2:

1. Drücken Sie [XEQ] ANN2 .
2. Geben Sie die Werte für die bekannten Variablen ein. Entspricht die Anzahl der Zahlungsperioden z.B. 60 (monatlich, über 5 Jahre), so tippen Sie 60 ein und drücken N .
3. Lösen Sie die Unbekannte durch Drücken der korrespondierenden Menütaste.
4. ANN2 benutzt *I%JR* zur Eingabeaufforderung und Anzeige des Zinssatzes (Jahreszinssatz, als Prozentsatz).
5. Als voreingestellter Wert für die Zahlungsperiode werden 12 Zahlungen pro Jahr verwendet, wobei als Voreinstellung für die Zahlweise *End-Modus* (Periodenende) dient. Um abweichende Werte zu spezifizieren, ist zuerst das MODE Menü aufzurufen; um dann z.B. 6 Zahlungen pro Jahr zu spezifizieren, drücken Sie einfach 6 #R/J .

   Sie können den *Beginn-Modus* festlegen, indem Sie BEG drücken.

   Durch Drücken von ANNU kehren Sie zum Hauptmenü zurück.

Beispiel: Algebraische Lösungen für finanzmathematische Aufgabenstellungen. In Kapitel 12 Ihres Benutzerhandbuchs ist die Höhe der monatlichen Zahlungen für einen Anschaffungskredit i.H.v. DM 15750,- bei einem nominalen Jahreszinssatz von 10,5% zu berechnen. Die Laufzeit des Kredits beträgt 3 Jahre, wobei End-Modus als Zahlweise definiert ist.

In diesem Beispiel ist ANNU auszuführen, was zu dem Wert *RATE* = −511,91 führt. ANNU benutzt zur Berechnung von *RATE* den *Löser*. Die Berechnung dauert etwa 3 Sekunden unter Verwendung der Anfangsnäherungen 0 und −500.

Teil 1. Verwenden Sie ANN2 zur expliziten Berechnung von *RATE*.

Stellen Sie als Anzeigeformat FIX 2 ein und führen Sie danach ANN2 aus.

■ DISP FIX 2 ENTER
XEQ ANN2

```
12 #R/J   END-MODUS
N  I%JR BARW RATE ENDW MODE
```

Geben Sie die bekannten Variablenwerte ein.

15750 BARW
10,5 I%JR
36 N
0 ENDW

```
ENDW=0,00
N  I%JR BARW RATE ENDW MODE
```

Berechnen Sie die monatliche Zahlung.

RATE

```
RATE=-511,91
N  I%JR BARW RATE ENDW MODE
```

Sie erhalten als Ergebnis −511,91 (gleich wie bei Anwendung von ANNU) bei einer Rechenzeit von weniger als eine Sekunde. Beachten Sie außerdem, daß die Rechenzeit *unabhängig* vom zuvor berechneten Wert für *RATE* ist. (Der Löser interpretiert den zuvor berechneten Wert als Anfangsnäherung, sofern nicht zwei Anfangsnäherungen vorgegeben wurden. Die explizite Lösung benutzt keine Anfangsnäherungen.)

Teil 2. Eine andere Bank bietet den gleichen Anschaffungskredit bei einer monatlichen Rückzahlungsrate von 530,- DM an. Welchen Jahreszinssatz (nominal) berechnet diese Bank?

530 +/- RATE
I%JR

```
I%JR=12,91
N  I%JR BARW RATE ENDW MODE
```

ANN2 verwendet den Löser zur Berechnung des neuen Zinssatzes. Der Löser benutzt die Anfangsnäherungen 0 und 20 (vom Programm vorgegeben) für den iterativen Lösungsprozeß; die Rechenzeit dauert etwa 11 Sekunden.

Verlassen Sie ANN2 und spezifizieren Sie wieder FIX 4 als Anzeigeformat.

EXIT
■ DISP FIX 4 ENTER

```
Y: 12,9104
X: 12,9104
```


Anwenden der SOLVE und PGMSLV Funktionen bei indirekter Adressierung

Im vorherigen Abschnitt wurde die SOLVE Funktion in ANN2 zur Berechnung des Zinssatzes *i* in der TVM Gleichung benutzt:

```
122 SOLVE "I%JR"
```

Die PGMSLV Funktion diente dabei zur Spezifikation der Routine, die die TVM Gleichung ausdrückt:

```
118 PGMSLV "i"
```

In ANN2 adressieren die SOLVE und PRGSLV Anweisungen *direkt* die Variablen und die Subroutine. Diese Art der direkten Adressierung erlaubt Ihnen das Spezifizieren von lediglich einer Löser-Routine, und innerhalb dieser Routine das Spezifizieren von nur einer Variablen. Allerdings läßt sich durch *indirekte Adressierung* die Anwendung des Lösers erweitern, was die Verwendung von mehreren Routinen und mehreren Variablen ermöglicht.

Beispiel: Anwenden von SOLVE für indirekte Adressierung. Betrachten Sie nochmals die Zustandsgleichung eines idealen Gases:

$$PV - nRT = 0$$

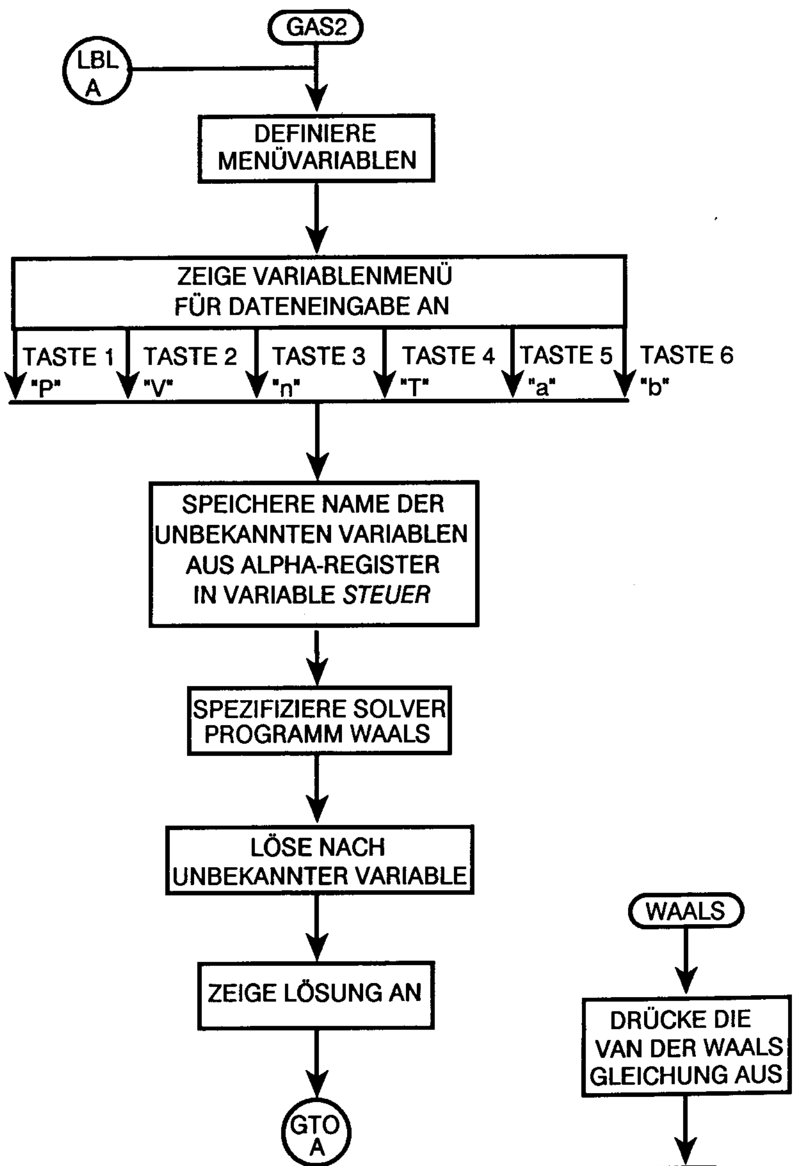
Die "Van der Waals" Zustandsgleichung modifiziert die Gleichung eines idealen Gases auf folgende Weise:

$$\left[\left(P + \frac{n^2 a}{V^2}\right)(V - nb)\right] - nRT = 0$$

wobei a und b Konstanten, charakteristisch für das entsprechende Gas, darstellen.

Teil 1. Schreiben Sie ein Programm, welches die Berechnung eines beliebigen Variablenwertes erlaubt, wobei entweder die Zustandsgleichung eines idealen Gases oder die Van der Waals'sche Gleichung verwendet wird.

Nachstehend finden Sie ein Flußdiagramm für das Programm, benannt als GAS2.



Nachstehend eine kommentierte Auflistung des Programms.

Programm:	**Kommentar:**
00 { 126-Byte Prgm } 01 LBL "GAS2"	
02 MVAR "P" 03 MVAR "V" 04 MVAR "n" 05 MVAR "T" 06 MVAR "a" 07 MVAR "b" 08 VARMENU "GAS2"	Zeile 02–08: Erzeuge das Variablenmenü.
09 CF 21 10 REALRES 11 STOP 12 ASTO "STEUER" 13 PGMSLV "WAALS" 14 SOLVE IND "STEUER" 15 VIEW IND "STEUER" 16 GTO "GAS2"	Zeile 09–16: Lösche Flag 21, um das Programm nach der VIEW Anweisung fortzusetzen. Nur reelle Lösungen zulassen. Zeige das Menü an und speichere den Namen der unbekannten Variablen in *STEUER* (Zeile 12). Spezifiziere als Löser-Routine "WAALS" (Zeile 13). *Indirektes* Adressieren der zu lösenden Variablen (Zeile 14). Zeige das Ergebnis an und springe zu Label GAS2 zurück (Zeile 15–16).
17 LBL "WAALS" 18 RCL "P" 19 RCL "n" 20 X↑2 21 RCL× "a" 22 RCL "V" 23 X↑2 24 ÷ 25 + 26 RCL "V" 27 RCL "n" 28 RCL× "b" 29 - 30 ×	Zeile 17–34, die Löser-Routine WAALS: Drücke die Van der Waals'sche Gleichung in der Weise aus, daß die rechte Seite Null ist.

```
31 0,082057
32 RCLx "n"
33 RCLx "T"
34 -

35 END
```

Teil 2. Verwenden Sie die Van der Waals'sche Zustandsgleichung zur Berechnung des Gasdrucks, der von 0,250 Mol Kohlendioxyd in 0,275 Liter bei 373 K ausgeübt wird; vergleichen Sie diesen Wert mit dem korrespondierenden Wert eines idealen Gases. Die Koeffizienten für CO_2 lauten: $a = 3{,}59\ l^2 - atm/mol^2$ und $b = 0{,}0427\ l/mol$.

Rufen Sie GAS2 auf.

XEQ GAS2

```
x: 0,0000
P V N T A B
```

Geben Sie die Werte für die bekannten Variablen ein.

,25 N
,275 V
373 T
3,59 A
,0427 B

```
b=0,0427
P V N T A B
```

Geben Sie für *P* die Anfangsnäherungen 10 und 30 ein und berechnen Sie *P*.

10 P
30 P
P

```
P=25,9816
P V N T A B
```

Unter Verwendung der Van der Waals'schen Gleichung ergibt sich als Druck der Wert 25,9816 atm.

Verwenden Sie nun die Zustandsgleichung für ein ideales Gas: Geben Sie einfach 0 für die Koeffizienten *a* und *b* vor und berechnen Sie *P*. Das vorherige Ergebnis für *P* dient als Anfangsnäherung.

0 A
0 B
P

```
P=27,8248
P V N T A B
```


Die Gleichung für ein ideales Gas führt zu einem Druck von 27,8248 Atmosphären. (Der tatsächlich gemessene Druck beträgt 26,1 atm.)

Verlassen Sie das Programm GAS2.

EXIT

```
Y: 27,8248
x: 27,8248
```

Einzelheiten zur Funktionsweise des Lösers

Nullstelle(n) einer Funktion

Wie Sie bereits wissen, muß zur Anwendung des Lösers ein Programm erzeugt werden, welches die Gleichung in der Art ausdrückt, daß die rechte Seite Null ist. Enthält die Gleichung mehrere Variablen, so sind nach dem Aufruf der SOLVER Applikation die Werte für alle bekannten Variablen vorzugeben. Zu diesem Zeitpunkt hat die Gleichung die Form $f(x) = 0$ angenommen, wobei x die Unbekannte und $f(x)$ das mathematische Kürzel für die *Funktion* ist, die x definiert. Betrachten Sie z.B. die Gleichung

$$2x^2 + xy + 10 = 3xz + 2yz$$

Um die Gleichung mit 0 gleichzusetzen, können Sie den Ausdruck auf der rechten Seite von beiden Seiten abziehen; dies führt zu der Form

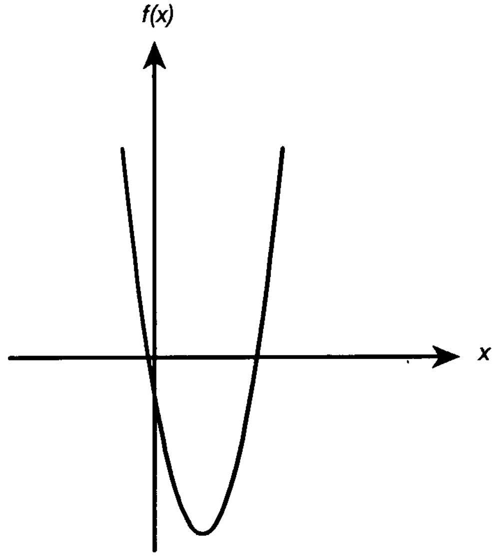
$$2x^2 + xy + 10 - 3xz - 2yz = 0$$

Zur Anwendung des Lösers ist nun ein Programm zu schreiben, welches die Variablen x, y und z definiert und die Gleichung ausdrückt. Rufen Sie nun die SOLVER Applikation auf und geben z.B. die Werte 2 für y und 3 für z ein, so ergibt sich für die Gleichung

$$2x^2 - 7x - 2 = 0$$

wobei x die unbekannte Variable ist und $f(x) = 2x^2 - 7x - 2$. Jeder Wert von x, für welchen sich $f(x) = 0$ ergibt, wird als *Nullstelle* der Funktion bezeichnet. Der Löser ermittelt iterativ eine Nullstelle für $f(x)$, indem die Funktion wiederholt für eine Näherung von x ausgeführt und das Ergebnis mit dem von vorangehenden Näherungen verglichen wird. Unter

Anwendung eines komplexen Algorithmus sagt der Löser eine neue Näherung für die Schnittstelle des Graphen von $f(x)$ mit der x-Achse voraus. Nachstehend eine Abbildung der Funktion $f(x) = 2x^2 - 7x - 2$. Wie daraus leicht zu erkennen ist, besitzt die Funktion zwei Nullstellen. (Das Beispiel auf Seite 110 – 112 berechnet diese Nullstellen.)



Von einer Ausnahme abgesehen stellen alle in diesem Abschnitt enthaltenen Beispiele die Funktion der Variable x dar. Denken Sie jedoch daran, daß die in den Beispielen beschriebenen Situationen sich ebenfalls auf Funktionen mit mehreren Variablen anwenden lassen, da diese Funktionen zu *einwertigen Funktionen werden*, wenn Sie in der SOLVER Applikation die Werte für die bekannten Variablen eingeben.

Fähigkeit des Lösers zum Auffinden einer Nullstelle

Damit der Löser eine Nullstelle auffinden kann, muß die Nullstelle innerhalb des Zahlenbereichs des Rechners liegen und $f(x)$ muß für den Wertebereich definiert sein, in welchem die iterative Suche stattfindet. Der Löser findet immer eine Lösung, wenn eine oder mehrere der nachstehenden Bedingungen erfüllt sind:

- Zwei Anfangsnäherungen führen zu $f(x)$ Werten mit entgegengesetzten Vorzeichen, und der Graph der Funktion schneidet die x-Achse wenigstens an einer Stelle zwischen diesen Näherungen (Abb. 3-1a).
- $f(x)$ nimmt immer zu oder ab, wenn x erhöht wird (Abb. 3-1b).
- Der Graph von $f(x)$ hat entweder überall eine konkave oder eine konvexe Form (Abb. 3-1c).
- $f(x)$ besitzt ein oder mehrere lokale Minima/Maxima und jedes tritt einzeln zwischen benachbarten Nullstellen von $f(x)$ auf (Abb. 3-1d).

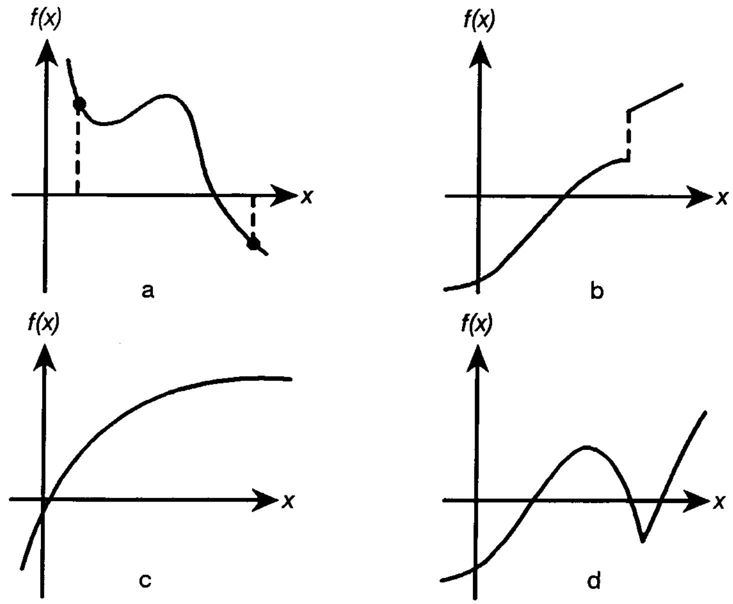


Abbildung 3-1: Funktionen mit lösbaren Nullstellen

In den meisten Situationen stellt die berechnete Nullstelle eine genaue Schätzung der theoretischen, unendlich genauen Nullstelle der Funktion dar. Eine *ideale* Lösung wäre vorhanden, wenn $f(x)$ genau gleich Null ist. Allerdings ist auch ein Wert ungleich Null für $f(x)$ ebenso oft akzeptabel, da dieser aus angenäherten Werten mit begrenzter (12-stelliger) Genauigkeit resultiert.

Interpretieren von Ergebnissen

Am Ende der iterativen Suche nach einer Nullstelle für die spezifizierte Funktion gibt der Löser Zahlenwerte in die Stackregister zurück. Unter vier Bedingungen wird zusätzlich eine Meldung angezeigt. Die Meldungen und Zahlenwerte können Ihnen bei der Interpretation des aufgefundenen Ergebnisses behilflich sein:

- Das X-Register enthält die bestmögliche Näherung. Dieser Wert *kann, muß jedoch nicht notwendigerweise*, die Nullstelle der Funktion darstellen.
- Das Y-Register enthält die vorangehende Näherung.
- Das Z-Register enthält den Funktionswert $f(x)$ für die bestmögliche Näherung.
- Das T-Register entält einen Code 0–4, welcher die Interpretation durch den Löser für das aufgefundene Ergebnis kennzeichnet. (Der Code wird im momentanen Anzeigeformat angezeigt: bei FIX 4 wird Code 0 als `0,0000` angezeigt.).

Code in T-Register	Interpretation	Meldung
0	Eine Lösung wurde gefunden.	
1	Der Löser verursachte einen Vorzeichenwechsel von $f(x)$ für benachbarte Werte von x, aber $f(x)$ divergierte sehr stark von 0, während x sich den Nachbarwerten von beiden Seiten nähert.	`Sign Reversal`
2	Der Löser hat eine Approximation für ein lokales Minimum/Maximum des numerischen Absolutbetrags gefunden. Ist $\pm 9{,}99999999999 \times 10^{499}$ die Lösung, so handelt es sich um einen asymptotischen Extremwert.	`Extremum`
3	Eine oder beide Anfangsnäherungen liegen außerhalb des Definitionsbereichs von $f(x)$. Das heißt, $f(x)$ gibt bei der Auswertung an diesen Stellen einen Fehler zurück.	`Bad Guess(es)`
4	$f(x)$ gibt für jede ausgewertete Näherung den gleichen Wert zurück.	`Constant?`

Im Fall einer aufgefundenen Nullstelle. Es gibt zwei Fälle, in welchen eine Nullstelle aufgefunden wird:

- Im Fall 1 ergibt sich für die gefundene Nullstelle $f(x)$ genau Null (Abbildung 3-2a).
- Im Fall 2 ergibt sich für die gefundene Nullstelle $f(x)$ nicht genau Null; sie stellt jedoch eine 12-stellige Zahl in unmittelbarer Nähe zum Schnittpunkt des Graphen mit der x-Achse (Abbildung 3-2b) dar. Dieser Fall tritt dann ein, wenn die letzten zwei Näherungen *benachbarte* Werte (sie unterscheiden sich um 1 in der 12. Stelle) darstellen und $f(x)$ ein unterschiedliches Vorzeichen für die Nachbarwerte annimmt. In den meisten Fällen liegt $f(x)$ relativ nahe bei 0.

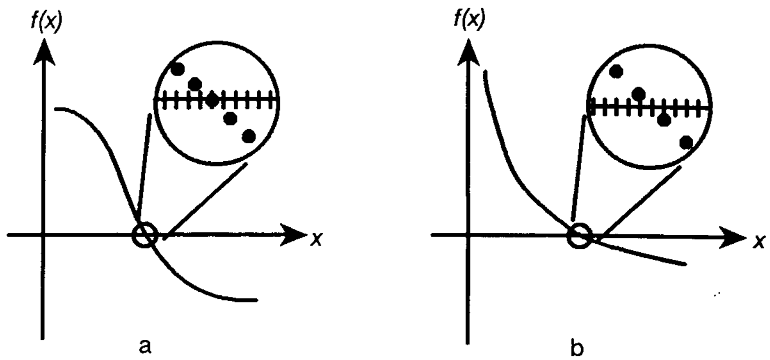


Abbildung 3-2: Fälle einer aufgefundenen Nullstelle

In beiden Fällen enthält das T-Register den Code 0 und es wird keine Meldung angezeigt. Sie können die beiden Fälle wie folgt unterscheiden:

- Ansehen des Z-Registerinhalts (der Funktionswert an der aufgefundenen Nullstelle). Bei einer Fall 2 Lösung handelt es sich um einen Wert ungleich 0.
- Vergleichen der bestmöglichen Näherung (Inhalt des X-Registers) und der vorangehenden Näherung (Inhalt des Y-Registers). Eine Fall 2 Lösung zeigt eine Abweichung um 1 in der 12. Stelle.
- Erneutes Lösen nach der Variablen. Bei einer Fall 2 Lösung zeigt der Löser die Meldung `Sign Reversal` nach dem zweiten Lösungsversuch an.

Beispiel: Eine Fall 1 Lösung mit zwei Nullstellen. Berechnen Sie die 2 Nullstellen der Gleichung:

$$2x^2 - 7x - 2 = 0$$

Drücken Sie die Funktion in Programm AA aus.

```
00 { 25-Byte Prgm }
01 LBL "AA"
02 MVAR "X"
03 RCL "X"
```



```
04 X↑2
05 2
06 ×
07 7
08 RCL× "X"
09 -
10 2
11 -
12 END
```

Spezifizieren Sie ALL als Anzeigeformat. Rufen Sie die SOLVER Applikation auf und wählen Sie danach Programm AA.

■ [DISP] ALL
■ [SOLVER]
AA

`x: 0`

Geben Sie die Anfangsnäherungen 1 und 5 für x ein und berechnen Sie x.

1 X
5 X
X

`X=3,76556443708`

Rollen Sie den Stackinhalt nach unten, um die vorangehende Näherung anzuzeigen.

[R↓]

`x: 3,76556443708`

Die Näherungswerte sind in allen 11 Nachkommastellen gleich. Rollen Sie den Stackinhalt weiter nach unten, um den Funktionswert an der Nullstelle anzuzeigen.

[R↓]

`x: 0`

$f(x)$ ist genau 0. Geben Sie nun die Anfangsnäherungen −0,1 und −1 für die zweite Nullstelle ein und berechnen Sie x.

,1 [+/-] X
1 [+/-] X
X

`X=-2,65564437075E-1`

Rollen Sie den Stackinhalt nach unten, um den Wert von $f(x)$ an der Nullstelle anzusehen. $f(x)$ entspricht wiederum genau 0.

[R↓] [R↓]

```
x: 0
```

Verlassen Sie den Löser und spezifizieren Sie wieder FIX 4 als Anzeigeformat.

[EXIT] [EXIT]
■[DISP] FIX 4 [ENTER]

```
Y: 0,0000
x: 0,0000
```

Beispiel: Eine Fall 2 Lösung. Im Beispiel auf Seite 101 – 105 berechneten Sie den Druck P in der Zustandsgleichung für ein ideales Gas durch Vorgabe der Werte für die restlichen Variablen V, n und T.

Berechnen Sie nun unter Verwendung der gleichen Werte für V, n und T den Wert von P.

Stellen Sie ALL als Anzeigeformat ein.

■[DISP] ALL

```
Select Solve Program
GAS2
```

Starten Sie Programm GAS2. (Geben Sie ggf. das Programm neu ein, falls Sie es zwischenzeitlich aus dem Programmspeicher gelöscht haben.)

[XEQ] GAS2

```
x: 0
 P  V  N  T  A  B
```

Geben Sie die bekannten Werte ein und berechnen Sie den Druck.

,25 N
,275 V
373 T
0 A
0 B
P

```
P=27,8247827273
 P  V  N  T  A  B
```

Rollen Sie den Stackinhalt nach unten, um die vorletzte Näherung anzusehen.

[R↓]

```
x: 27,8247827272
 P  V  N  T  A  B
```


Die Näherungen unterscheiden sich um 1 in der letzten Dezimalstelle. Rollen Sie den Stack nochmals nach unten, um den Wert für $f(x)$ anzusehen.

[R↓]

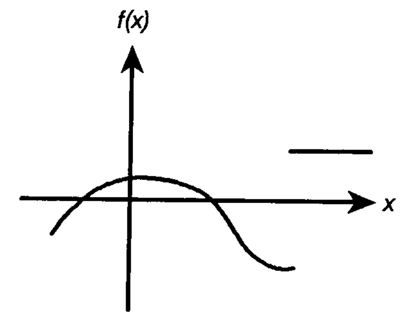
```
x: 0,00000000001
 P  V  N  T  A  B
```

Der Funktionswert an der Näherung ist eine sehr kleine Zahl ungleich 0. Es liegt keine genaue Nullstelle vor, es handelt sich jedoch um eine sehr gute Näherung. Verlassen Sie das Programm und stellen Sie wieder FIX 4 als Anzeigeformat ein.

[EXIT]
■[DISP] FIX 4 [ENTER]

```
Y: 0,0000
x: 1,0000E-11
```

Besondere Problemstellungen. Einige Arten von Funktionen erfordern besondere Überlegungen, bevor eine Lösung aufgefunden werden kann. Betrachten Sie z.B. die nachstehende Funktion, welche eine Unstetigkeitsstelle beim Schneiden der x-Achse besitzt.



Der Löser ermittelt einen x-Wert in der Nachbarschaft zur Unstetigkeitsstelle; der Wert von $f(x)$ kann dabei relativ groß sein.

Beispiel: Unstetige Funktion. Berechnen Sie die Nullstelle der Gleichung

$$IP(x) - 1{,}5 = 0$$

Drücken Sie die Funktion in Programm BB aus.

```
00 { 18-Byte Prgm }
01 LBL "BB"
02 MVAR "X"
03 RCL "X"
04 IP
05 1,5
06 -
07 END
```

Rufen Sie die SOLVER Applikation auf und wählen Sie das Programm BB; geben Sie danach die Anfangsnäherungen 0 und 5 ein und lösen Sie nach x.

■ [SOLVER] BB
0 X
5 X
X

```
X=2,0000
```

Der Löser ermittelt eine Nullstelle bei $x = 2{,}0000$. Überprüfen Sie nun den Wert von $f(x)$.

[R↓] [R↓]

```
x: -0,5000
```

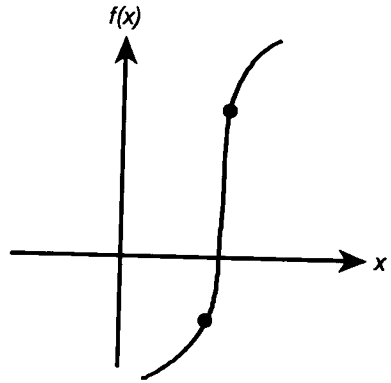
Der Wert von $f(x)$ erscheint relativ groß. Dies deutet darauf hin, daß Sie die Funktion näher auswerten sollten. Eine Abbildung der Funktion läßt erkennen, daß bei $x = 2{,}0000$ in Wirklichkeit eine Unstetigkeitsstelle und keine Nullstelle vorliegt.

Verlassen Sie den Löser.

[EXIT] [EXIT]

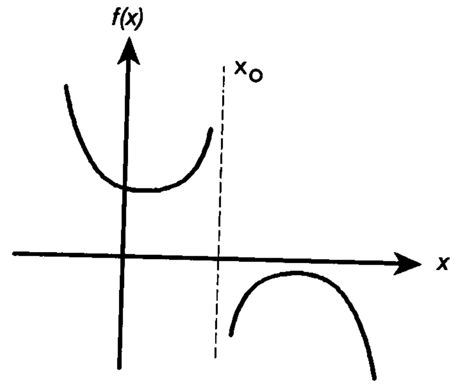
```
Y: 0,0000
x: -0,5000
```

Betrachten Sie als Abschluß die nachstehende Funktion. Sie hat eine sehr große Steigung in der Umgebung der Nullstelle. Eine Auswertung benachbarter Werte kann zu relativ großen Werten führen, obwohl es sich um eine wahre Nullstelle zwischen den benachbarten Werten handelt.



Der Interpretation von Löser-Ergebnissen ist demnach etwas Beachtung zu widmen. Der Löser ist am effektivsten, wenn Sie ihn in Verbindung mit Ihrer eigenen Analyse der auszuwertenden Funktion einsetzen.

Ein Vorzeichenwechsel. Die Werte der folgenden Funktion streben an der Stelle x_0 gegen unendlich, wobei der Graph das Vorzeichen wechselt.



Die Funktion hat einen *Pol* an der Stelle x_0. Bei der Auswertung solch einer Funktion gibt der Löser die Meldung `Sign Reversal` zurück.

Beispiel: Ein Pol. Berechnen Sie die Nullstelle der Gleichung

$$\frac{x}{(x^2 - 6)} - 1 = 0$$

Bei der Näherung von x an $\sqrt{6}$ nimmt $f(x)$ einen sehr großen positiven oder negativen Betrag an.

Drücken Sie die Funktion in Programm CC aus.

```
00 { 23-Byte Prgm }
01 LBL "CC"
02 MVAR "X"
03 RCL "X"
04 RCL "X"
05 X↑2
06 6
07 -
08 ÷
09 1
10 -
11 END
```

Rufen Sie die SOLVER Applikation auf und wählen Sie das Programm CC.

■ [SOLVER] CC

```
x: 0,0000
 X
```

Geben Sie die Anfangsnäherungen 2,3 und 2,7 vor und berechnen Sie x.

2,3 X
2,7 X
X

```
X=2,4495
Sign Reversal
```

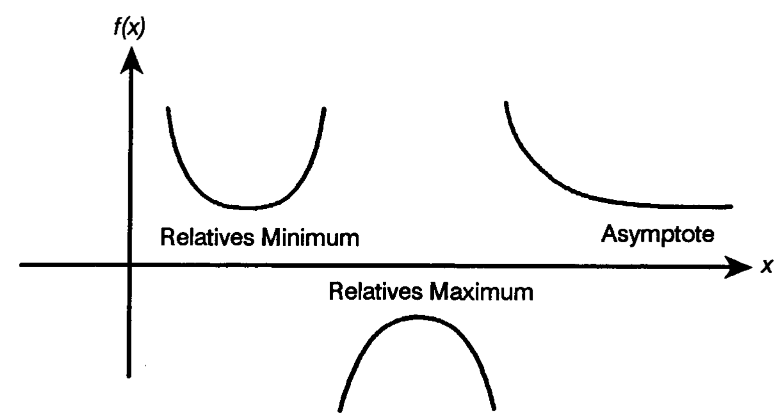

Die Anfangsnäherungen führten zu unterschiedlichen Vorzeichen für $f(x)$. Das Intervall zwischen aufeinanderfolgenden Näherungen wurde verkleinert, bis zwei Nachbarwerte gefunden wurden. Für diese nähert sich $f(x)$ jedoch einem Pol anstatt einer Nullstelle. Die Funktion *besitzt* Nullstellen bei −2 und 3, welche durch Vorgabe besserer Anfangsnäherungen gefunden werden können.

Verlassen Sie den Löser.

[EXIT] [EXIT]

```
Y: 2,4495
x: 2,4495
```

Extrempunkte. Wenn der Löser die Meldung `Extremum` anzeigt, wurde eine Approximation eines lokalen Minimums oder Maximums für den Absolutbetrag der Funktion gefunden. Liegt als Lösung (der Wert im X-Register) $\pm 9{,}99999999999 \times 10^{499}$ vor, so hat der Löser einen *asymptotischen* Extrempunkt gefunden.



Beispiel: Ein relatives Minimum. Berechnen Sie die Lösung der Parabel-Gleichung

$$x^2 - 6x + 13 = 0$$

(Sie hat ein Minimum bei $x = 3$.)

Drücken Sie die Funktion im Programm DD aus.

```
00 { 23-Byte Prgm }
01 LBL "DD"
02 MVAR "X"
03 RCL "X"
04 X↑2
05 6
06 RCL× "X"
07 -
08 13
09 +
10 END
```

Rufen Sie die SOLVER Applikation auf und danach das Programm DD.

■[SOLVER] DD

```
x: 0,0000
X
```

Geben Sie die Anfangsnäherungen 0 und 10 ein und lösen Sie nach x.

0 X
10 X
X

```
X=3,0000
Extremum
```

Verlassen Sie den Löser.

[EXIT] [EXIT]

```
y: 3,0000
x: 3,0000
```


Beispiel: Eine Asymptote. Ermitteln Sie die Lösungen für die Gleichung

$$10 - \frac{1}{x} = 0$$

Drücken Sie die Funktion in Programm EE aus.

```
00 { 17-Byte Prgm }
01 LBL "EE"
02 MVAR "X"
03 10
04 RCL "X"
05 1/X
06 -
07 END
```

Rufen Sie die SOLVER Applikation auf und danach das Programm EE.

■[SOLVER] EE

```
x: 0,0000
X
```

Geben Sie die Anfangsnäherungen 0,005 und 5 vor und lösen Sie nach x.

,005 X
5 X
X

```
X=0,1000
X
```

Der Löser ermittelt eine Nullstelle bei $x = 0{,}1000$. Geben Sie nun negative Werte als Anfangsnäherungen vor.

1 [+/-] X
2 [+/-] X
X

```
X=-1,0000E500
Extremum
```

Der Löser ermittelt einen asymptotischen Extremwert. (Drücken Sie ■[SHOW] zum Überprüfen, ob als Lösung tatsächlich $-9{,}99999999999 \times 10^{499}$ vorliegt.) Bei näherer Betrachtung der Gleichung ist ersichtlich, daß sich für negative x-Werte als kleinster Funktionswert 10 ergeben kann; $f(x)$ *strebt* für große negative x-Werte *asymptotisch* gegen 10.

Verlassen Sie den Löser.

[EXIT] [EXIT]

```
y: -5,9246E498
x: -1,0000E500
```

Schlechte Anfangsnäherung(en). Der Löser gibt die Meldung `Bad Guess(es)` aus, wenn eine oder beide Anfangsnäherungen außerhalb des Definitionsbereichs der Funktion liegen. (Bei der Auswertung für die Anfangsnäherung führt die Funktion zu einem mathematisch bedingten Fehler.)

Beispiel: Mathematisch bedingter Fehler. Berechnen Sie die Nullstelle der Gleichung

$$\sqrt{\frac{x}{(x + 0{,}3)}} - 0{,}5 = 0$$

Drücken Sie die Funktion in Programm FF aus.

```
00 { 26-Byte Prgm }
01 LBL "FF"
02 MVAR "X"
03 RCL "X"
04 0,3
05 RCL+ "X"
06 ÷
07 SQRT
08 0,5
09 -
10 END
```

Rufen Sie die SOLVER Applikation auf und wählen Sie das Programm FF.

■ SOLVER FF

```
x: 0,0000
 X
```

Versuchen Sie zuerst, über die Anfangsnäherungen 0 und 10 eine positive Nullstelle aufzufinden.

0 X
10 X
X

```
X=0,1000
 X
```


Der Löser findet eine Nullstelle bei $x = 0{,}1$. Versuchen Sie nun, eine negative Nullstelle unter Verwendung der Anfangsnäherungen $-0{,}1$ und $-0{,}2$ aufzufinden. Beachten Sie, daß die Funktion für Werte von x zwischen 0 und $-0{,}3$ nicht definiert ist, da diese Werte zu einem positiven Nenner und einem negativen Zähler führen, was eine negative Quadratwurzel ergibt. Obwohl der HP-42S arithmetische Operationen mit komplexen Zahlen ausführen kann, betrachtet der Löser die Funktion an der ausgewerteten Stelle als undefiniert, wenn ein komplexes Ergebnis erhalten wurde.

,1 +/- X
,2 +/- X
X

```
X=-0,2000
Bad Guess(es)
```

Verlassen Sie den Löser.

EXIT EXIT

```
Y: -0,1000
x: -0,2000
```

Eine Konstante. Der Löser zeigt die Meldung `Constant?` an, wenn die Auswertung von $f(x)$ immer zu gleichen Funktionswerten führt. Diese Situation kann eintreten, wenn Anfangsnäherungen auf einen lokalen "flachen" Bereich einer Funktion beschränkt sind.

Beispiel: Lokaler flacher Bereich. Berechnen Sie die Nullstelle der Gleichung

$$\frac{1}{x} - 10 = 0$$

Drücken Sie die Funktion im Programm GG aus.

```
00 { 17-Byte Prgm }
01 LBL "GG"
02 MVAR "X"
03 RCL "X"
04 1/X
05 10
06 -
07 END
```

Rufen Sie den Löser auf und wählen Sie das Programm GG.

■[SOLVER] GG

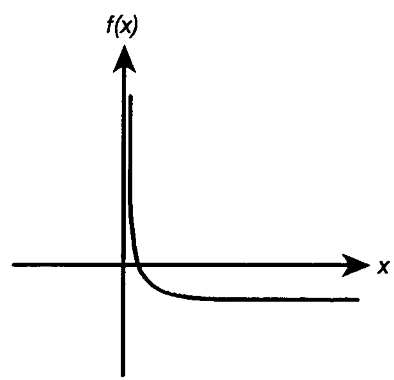
```
x: 0,0000
```

Geben Sie die Anfangsnäherungen 10^{20} und 10^{30} vor.

[E] 20 X
[E] 30 X
X

```
X=1,0000E500
Constant?
```

Der Wert von $f(x)$ ist für diesen Bereich immer gleich (bei gegebener 12-stelliger Genauigkeit durch den Rechner). Nachstehend finden Sie eine Abbildung der Funktion.



Versuchen Sie es mit den Anfangsnäherungen 0 und 10.

0 X
10 X
X

```
X=0,1000
```

Der Löser berechnet die Nullstelle $x = 0{,}1$. Verlassen Sie den Löser.

[EXIT] [EXIT]

```
Y: 0,1000
x: 0,1000
```


Rundungsfehler und "Underflow"

Rundungsfehler. Die (begrenzte) 12-stellige Genauigkeit des Rechners ist in fast allen Anwendungsfällen ausreichend. Es gibt jedoch auch Ausnahmen, wo ein Rundungsfehler ein Löser-Ergebnis beeinflußt. Z.B. hat die Gleichung

$$[(|x| + 1) + 10^{15}]^2 - 10^{30} = 0$$

keine Nullstellen, da $f(x)$ immer positiv ist; der Löser ermittelt allerdings über die Anfangsnäherungen 1 und 2 als Ergebnis 1,0000, was auf Rundungsfehler zurückzuführen ist.

Rundungsfehler können den Löser auch dazu veranlassen, keine Nullstelle aufzufinden. Die Gleichung

$$|x^2 - 7| = 0$$

hat eine Nullstelle bei $\sqrt{7}$. Allerdings kann $\sqrt{7}$ mit 12-stelliger Genauigkeit *nicht exakt* dargestellt werden. Außerdem wechselt die Funktion nie das Vorzeichen. Der Löser gibt als Hinweis die Meldung `Extremum` aus. Die Endnäherung von x stellt jedoch die bestmögliche 12-stellige Approximation der Nullstelle dar, wenn die Routine den Lösungsprozeß abbricht.

"Underflow" . Ein *Bereichsunterlauf* kann eintreten, wenn der Betrag einer Zahl kleiner als die kleinste im Rechner darstellbare Zahl ist, wodurch eine Substitution durch Null erfolgt. Dieser Umstand kann ebenfalls die Ergebnisse der SOLVER Applikation beeinflussen. Betrachten Sie z.B. die Gleichung

$$\frac{1}{x^2} = 0$$

Die Nullstelle dieser Gleichung ist unendlich. Aufgrund eines Bereichsunterlaufs gibt der Löser eine sehr große (endliche) Zahl zurück. (Davon abgesehen kann der Rechner den Wert "unendlich" nicht darstellen.)

4

Integration

Dieses Kapitel geht auf folgende Themen ein:

- Normale Anwendung der Integrationsapplikation.
- Approximation eines Integrals mit unendlicher oberer oder unterer Integrationsgrenze.
- Interaktive Anwendung von SOLVER und Integration.
- Einzelheiten zur Funktionsweise des Integrationsalgorithmus.

Normale Integration

Zur Ausführung der Integrationsapplikation ist wie folgt vorzugehen:

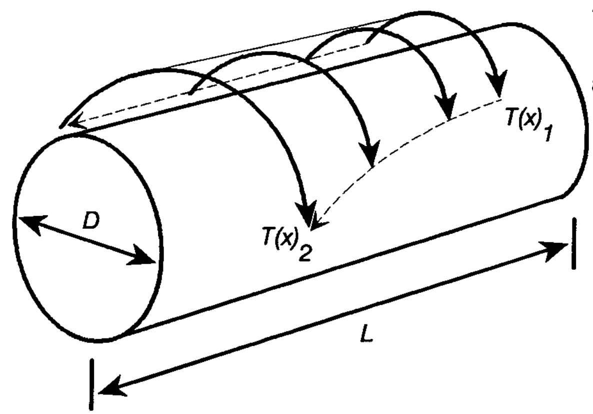
1. Erzeugen Sie ein Programm, welches:
   - **a.** MVAR zur Definition der Variable(n) im Integranden (die zu integrierende Funktion) verwendet.
   - **b.** Den Integrand ausdrückt. (Beachten Sie, daß jede Variable im Integrand in das X-Register zurückgerufen werden muß.)
2. Wenden Sie die Integrationsapplikation auf das Programm an.
   - **a.** Wählen Sie die Integrationsapplikation (drücken Sie ■ [∫f(x)]).
   - **b.** Wählen Sie das gewünschte Programm durch Drücken der korrespondierenden Menütaste.
   - **c.** Geben Sie die Werte für jede bekannte Variable ein. Wählen Sie die Integrationsvariable.
   - **d.** Geben Sie die Werte für *LLIM*, *ULIM* und *ACC* ein.
   - **e.** Drücken Sie [∫], um mit der Berechnung zu beginnen.

Beispiel: Einfache Integration. Der Torsionswinkel in einer Welle kann durch das folgende Integral berechnet werden:

$$\theta = \int_0^L \frac{T}{JG}\,dx$$

wobei:

θ = Torsionswinkel der Welle (in Radiant).
L = Länge der Welle (in m).
T = auf Welle ausgeübtes Drehmoment (in Nm).
J = Flächenträgheitsmoment (in m^4).
G = Scherungsmodul (in N/m^2).



Torsion *T* vergrößert sich über die Wellenlänge als eine Funktion von *x*.

Betrachten Sie eine massive Welle aus Stahl ($G = 83 \times 10^9\ N/m^2$), die einen Durchmesser von 0,03 Meter ($J = 7{,}9521 \times 10^{-8}\ m^4$) und eine Gesamtlänge L von 2 Meter besitzt. Berechnen Sie den Torsionswinkel für die Welle, wenn auf diese ein Drehmoment ausgeübt wird, welches sich entlang der Länge x als Funktion von x verhält:

$$T = 13x^4 + 8x^3 + 15x^2 + 9x + 6$$

Verwenden Sie aus Programmierungszwecken das Hornersche Schema, um das Polynom zu erweitern.

$$T = (((13x + 8)x + 15)x + 9)x + 6$$

Durch Substituieren dieses Ausdrucks für T erhalten Sie die Gleichung

$$\theta = \int_0^L \frac{(((13x+8)x+15)x+9)x+6}{JG}\,dx$$

Drücken Sie den Integranden im Programm TORSION aus.

Programm:	**Kommentar:**
`00 { 54-Byte Prgm }`	
`01 LBL "TORSION"`	
`02 MVAR "X"`	Zeile 02–04: Definiere die Variablen.
`03 MVAR "J"`	
`04 MVAR "G"`	
`05 13`	Zeile 05–19: Drücke den Integranden aus.
`06 RCL× "X"`	
`07 8`	
`08 +`	
`09 RCL× "X"`	
`10 15`	
`11 +`	
`12 RCL× "X"`	
`13 9`	
`14 +`	
`15 RCL× "X"`	
`16 6`	
`17 +`	
`18 RCL÷ "J"`	
`19 RCL÷ "G"`	
`20 END`	

Rufen Sie die Integrationsapplikation auf.

■ [∫f(x)]

```
Select ∫f(x) Program
TORSI
```

Wählen Sie das Programm TORSION.

TORSI

```
Set Vars; Select ∫var
X  J  G
```


Geben Sie die bekannten Werte für J und G ein und spezifizieren Sie die Integrationsvariable X.

7,9521 [E] 8 [+/-] J
83 [E] 9 G
X

```
x: 83.000.000.000,0
LLIM ULIM ACC ∫
```

Spezifizieren Sie die untere (0) und obere Integrationsgrenze L (2) sowie den Genauigkeitsfaktor 0,01.

0 LLIM
2 ULIM
,01 ACC

```
ACC=0,0100
LLIM ULIM ACC ∫
```

Starten Sie die Berechnung.

∫

```
∫=0,0281
LLIM ULIM ACC ∫
```

Es ergibt sich für die Welle ein Torsionswinkel θ = 0,0281 Radiant (1,6077 Grad). Verlassen Sie die Integrationsapplikation.

[EXIT] [EXIT] [EXIT]

```
y: 0,0003
x: 0,0281
```

Approximation eines Integrals mit unendlicher Integrationsgrenze

Bestimmte Aufgabenstellungen erfordern die Auswertung eines *unbestimmten Integrals*, welches eine unendliche obere oder untere Integrationsgrenze besitzt. Ein unbestimmtes Integral mit einer unendlichen Obergrenze

$$\int_0^\infty f(x)\,dx$$

wird "von Hand" berechnet, indem der folgende gleichwertige Ausdruck ausgewertet wird:

$$\lim_{a\to\infty} \int_0^a f(x)\,dx$$

Sie können den HP-42S nicht zur direkten Auswertung solch eines Ausdrucks verwenden. Sie können jedoch durch Substitution der unendlichen Grenze für eine sehr große Zahl eine *Approximation* berechnen.

Beispiel: Auswerten eines Integrals mit unendlicher Obergrenze. Berechnen Sie das Integral

$$\int_0^{\infty} \frac{dx}{1+x^2}$$

von Hand. Bestimmen Sie anschließend eine Approximation des Integrals über den HP-42S.

Teil 1. Sie erhalten das manuelle Ergebnis wie folgt:

$$\int_0^{\infty} \frac{dx}{1+x^2} = \lim_{a \to \infty} \int_0^{a} \frac{dx}{1+x^2}$$
$$= \lim_{a \to \infty} (\arctan a)$$
$$= \frac{\pi}{2}$$

Verwenden Sie den HP-42S zur Berechnung von $\pi/2$ mit 12-stelliger Genauigkeit.

■ [DISP] ALL
■ [π] 2 [÷]

```
Y: 0
X: 1,5707963268
```

Teil 2. Verwenden Sie die Integrationsapplikation zur Auswertung des gleichen Integrals, wobei der Wert 1000 zur Approximation der Obergrenze einzusetzen ist. Drücken Sie zuerst den Integranden im Programm INFIN aus.

```
00 { 20-Byte Prgm }
01 LBL "INFIN"
02 MVAR "X"
03 RCL "X"
04 X↑2
05 1
06 +
07 1/X
08 END
```


Rufen Sie die Integrationsapplikation und danach das Programm INFIN auf.

■ [∫f(x)] INFIN

```
Set Vars; Select ∫var
X
```

Wählen Sie die Integrationsvariable.

X

```
x: 1,5707963268
LLIM ULIM ACC ∫
```

Spezifizieren Sie die Untergrenze (0), die Approximation für die Obergrenze (1000) und den Genauigkeitsfaktor 0,01.

0 LLIM
1000 ULIM
,01 ACC

```
ACC=0,01
LLIM ULIM ACC ∫
```

Berechnen Sie das Integral.

∫

```
∫=1,57020935993
LLIM ULIM ACC ∫
```

Sie erhalten als Ergebnis 1,57020935993, was bis zu 3 Dezimalstellen korrekt ist. Als Rechenzeit wurden etwa 36 Sekunden beansprucht.

Verlassen Sie die Integrationsapplikation und stellen Sie wieder FIX 4 als Anzeigeformat ein.

[EXIT] [EXIT] [EXIT]
■ [DISP] FIX 4 [ENTER]

```
Y: 0,0156
X: 1,5702
```

Die nachstehende Tabelle faßt Ergebnisse und Rechenzeiten von Approximationen der Obergrenze für 100, 1000 und 10000 sowie von Genauigkeitsfaktoren 0,01 und 0,0001 zusammen.

ACC Faktor	ULIM	Ergebnis	Rech.zeit (Sekunden)
		1,5707963268 (tats. $\frac{\pi}{2}$)	
0,01	100	1,57518831857	5
	1000	1,57020935993	36
	10000	1,57088603739	140
0,0001	100	1,5607891695	18
	1000	1,56979476064	69
	10000	1,57069673168	279

Beachten Sie, daß der wesentliche Faktor bei der Bestimmung der Genauigkeit des Ergebnisses der Wert der Approximation für die Obergrenze ist, nicht der Genauigkeitsfaktor. Außerdem ist anzumerken, daß die Berechnungen mit einem Genauigkeitsfaktor von 0,0001 etwa doppelt solang dauern als Berechnungen mit einem Genauigkeitsfaktor von 0,01.

Im allgemeinen sollten Sie beim Festlegen der Genauigkeit für die Approximation eines Integrals die von Ihnen gewählte Approximation der Integrationsgrenze mit berücksichtigen. Führt die substituierte Grenze nur zu einer groben Approximation für das wahre Integral, so ist es nicht sehr sinnvoll, diese Approximation mit einer großen Genauigkeit (kleiner Genauigkeitsfaktor) zu berechnen.

Interaktive Anwendung von SOLVER und Integration

Im ersten Beispiel dieses Kapitels berechneten Sie den Torsionswinkel θ am Ende einer Welle, indem Sie das ausgeübte Drehmoment über x integriert haben. (Das Drehmoment verhält sich wie eine Funktion von der Position x entlang der Welle.) In diesem Beispiel war die Einschränkung gegeben, speziell den Torsionswinkel θ zu berechnen. Für die Gleichung

$$I = \int_{LLIM}^{ULIM} f(x)\,dx \text{ (berechnet mit Genauigkeit } ACC)$$

erlaubt die Integrationsapplikation im allgemeinen *nur* das Berechnen der Variablen I in der Gleichung. Um I zu berechnen:

- Schreiben Sie ein Programm P, das den Integranden $f(x)$ definiert.
- Spezifizieren Sie die Werte der bekannten Variablen im Integranden.
- Spezifizieren Sie die Integrationsvariable.
- Spezifizieren Sie die Werte für die Variablen *LLIM*, *ULIM* und *ACC*.

Allerdings können Sie durch das Schreiben eines Programms S für SOLVER jede beliebige Variable der Gleichung lösen:

- I
- Die Variablen im Integranden $f(x)$.
- *LLIM*, *ULIM*.

S muß dabei jede Variable der Gleichung definieren und *die Integrationsapplikation auf Programm P* anwenden. Im nachstehenden Beispiel ist die Länge L einer Welle (die Variable *ULIM* in der Integrationsapplikation) in der Torsionswinkel-Gleichung zu berechnen.

Beispiel. Interaktive Anwendung von Löser und Integration. Nachstehend nochmals die Gleichung zur Berechnung des Torsionswinkels in einer Welle:

$$\theta = \int_0^L \frac{T}{JG}\,dx$$

Betrachten Sie nochmals die Stahlwelle aus dem ersten Beispiel dieses Kapitels. Für diese Welle wurden als Variablenwerte angenommen: $G = 83 \times 10^9$ N/m² und $J = 7{,}9521 \times 10^{-8}$ m⁴. Die Welle wird der gleichen Torsionsbeanspruchung T wie im ersten Beispiel ausgesetzt, wobei sich T entlang der Wellenlänge x wie $f(x)$ verhält.

$$T = 13x^4 + 8x^3 + 15x^2 + 9x + 6.$$

Berechnen Sie die Länge L, welche zu einem Torsionswinkel θ von 0,1396 Radiant (8 Grad) führt.

Die Gleichung enthält die Variablen θ, L, T, J und G. Die unbekannte Variable L stellt dabei die obere Integrationsgrenze *ULIM* dar.

Teil 1. Schreiben Sie das Löser-Programm WELLE, welches:

- Jede in der Gleichung enthaltene Variable definiert.
- Die Gleichung so ausdrückt, daß die rechte Seite Null ist.

$$\int_0^L \frac{T}{JG}\,dx - \theta = 0$$

Programm:	**Kommentar:**
`00 { 61-Byte Prgm }`	
`01 LBL "WELLE"`	
`02 MVAR "THETA"`	Zeile 02 – 08: Definiere die Variablen.
`03 MVAR "G"`	
`04 MVAR "J"`	
`05 MVAR "LLIM"`	
`06 MVAR "ULIM"`	
`07 MVAR "ACC"`	
`08 MVAR "X"`	
`09 PGMINT "TORSION"`	Zeile 09 – 11: Drücke Gleichung so aus, daß ihre rechte Seite Null entspricht. Berechne zuerst den ersten Term der Gleichung (das Integral) (Zeile 09 – 10). Der Wert des Integrals wird in das X-Register zurückgegeben. Subtrahiere den 2. Term (*THETA*).
`10 INTEG "X"`	
`11 RCL- "THETA"`	
`12 END`	

In Zeile 09 – 10 wird das Integral unter Verwendung des momentanen Wertes von *ULIM*, welcher iterativ vom Löser während der Suche nach einer Lösung vorgegeben wird, berechnet. Beachten Sie, daß das spezifizierte Programm zur Integration dem Programm aus dem ersten Beispiel (TORSION) entspricht. Falls Sie dieses Programm zwischenzeitlich gelöscht haben, ist dies nun erneut in den Rechner einzugeben. test

Teil 2. Wählen Sie die SOLVER Applikation und danach das Programm WELLE.

■ [SOLVER] WELL

```
x: 0,0000
THETA  G  J  LLIM  ULIM  ACC
```

(Die Variable X ist in der zweiten Menüzeile enthalten.) Geben Sie die Werte für die bekannten Variablen ein.

,1396 THETA
83 [E] 9 G
7,9521 [E] 8 [+/-] J
0 LLIM
,01 ACC

```
ACC=0,0100
THETA  G  J  LLIM  ULIM  ACC
```

Berechnen Sie nun die obere Integrationsgrenze L unter Vorgabe der Anfangsnäherungen 1 und 10.

1 ULIM
10 ULIM
ULIM

```
ULIM=2,9528
THETA  G  J  LLIM  ULIM  ACC
```

Es ergibt sich eine Wellenlänge von 2,9528 Meter.

Verlassen Sie die SOLVER Applikation.

[EXIT] [EXIT]

```
y: 2,9528
x: 2,9528
```

Einzelheiten zur Funktionsweise des Integrationsalgorithmus

Genauigkeitsfaktor und Fehlerabschätzung für Integration

Der Integrationsalgorithmus berechnet das Integral einer Funktion $f(x)$, indem er einen gewichteten Mittelwert der Funktionswerte an ausreichend vielen Stützstellen von x innerhalb des Integrationsintervalls bildet. Die Genauigkeit des Ergebnisses hängt hierbei von der Anzahl der einbezogenen Stützstellen ab. Allgemein gilt: je mehr Stützstellen, desto größer die Genauigkeit. Es gibt zwei Gründe, die eine Einschränkung der Genauigkeit des Integrals erforderlich machen können:

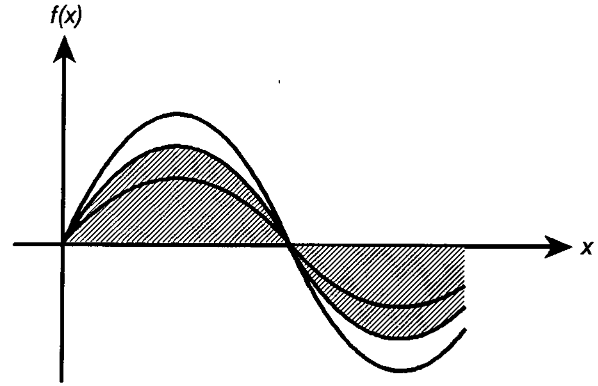
1. Die Rechenzeit nimmt mit zunehmender Anzahl von Stützstellen zu.
2. Die Genauigkeit des Integranden selbst hängt wiederum von drei Betrachtungen ab:
   - **a.** *Die Genauigkeit von empirisch abgeleiteten* Konstanten in $f(x)$. Wenn z.B. $f(x)$ empirische Konstanten enthält, welche nur bis zu 2 Dezimalstellen genau sind, so ist es nicht sehr sinnvoll, das Integral mit der vollen Genauigkeit des Rechners (12 Stellen) zu berechnen.
   - **b.** Bis zu welchem Grad kann $f(x)$ einen physikalischen Vorgang genau beschreiben?
   - **c.** Welches Ausmaß nehmen die Rundungsfehler bei der internen Auswertung des Ausdrucks $f(x)$ an?

Um die Genauigkeit des *Integrals* indirekt einzuschränken, ist der *Genauigkeitsfaktor* anhand folgender *Funktion* zu spezifizieren:

$$ACC = \left| \frac{\text{wahrer Wert von } f(x) - \text{berechneter Wert von } f(x)}{\text{berechneter Wert von } f(x)} \right|$$

Der Genauigkeitsfaktor stellt Ihre Schätzung des prozentualen Fehlers (als Dezimalwert) für jeden berechneten Wert von $f(x)$ dar. Dieser Wert ist in *ACC* gespeichert. Er bezieht sich auf die *Fehlerabschätzung für die Integration* in der Form:

$$\textit{Fehlerabschätzung für Integration} = \textit{Genauigkeitsfaktor} \times \int |\, f(x) \,|\, dx$$



Die schraffierte Fläche stellt den Wert des Integrals dar. Die Fläche, welche durch die obere und untere Kurve gebildet wird, ergibt sich aus der gewichteten Summe der Fehler für jede Berechnung von $f(x)$. Sie können erkennen, daß für jeden Punkt x die Fehlerabschätzung proportional zu $f(x)$ ist.

Der Integrationsalgorithmus verwendet ein iteratives Verfahren, wobei die Anzahl der Stützstellen bei jedem folgenden Iterationsschritt verdoppelt wird. Am Ende jeder Iteration wird das Integral und die Fehlerabschätzung der Integration berechnet. Anschließend erfolgt ein Vergleich des soeben berechneten Integralwerts mit den Werten aus zwei vorhergehenden Werten. Ist der Unterschied zwischen einem dieser drei Werte und den anderen beiden kleiner als die Fehlerabschätzung für die Integration, so wird der Iterationsprozeß abgebrochen. Der momentane Wert des Integrals wird in das X-Register und die zugehörige Fehlerabschätzung in das Y-Register zurückgegeben.

Es ist sehr unwahrscheinlich, daß die Fehler in drei aufeinanderfolgenden Berechnungen des Integrals – d.h. der Unterschied zwischen dem tatsächlichen Integral und den berechneten Werten – immer größer als der

Unterschied zwischen den berechneten Werten selbst ist. Deshalb ist der Fehler des zuletzt berechneten Werts fast mit Sicherheit kleiner als die Fehlerabschätzung der Integration.

Beispiel: Genauigkeitsfaktor und Fehlerabschätzung für Integration. In der Nachrichtentechnik wird für manche Zwecke (z.B. die Stromübertragung in idealisierten Netzwerken) ein Integral der folgenden Form (auch *Integralsinus* genannt) benötigt:

$$\mathrm{Si}(t) = \int_0^t \frac{\sin x}{x}\, dx$$

Berechnen Sie Si (2).

Schreiben Sie zuerst ein Programm, welches die Funktion ausdrückt.

```
00 { 16-Byte Prgm }
01 LBL "SI"
02 MVAR "X"
03 RCL "X"
04 SIN
05 RCL÷ "X"
06 END
```

Spezifizieren Sie ALL als Anzeigeformat und RAD als Winkelmodus.

■ DISP ALL
■ MODES RAD

```
Y: 0
X: 0
```

Wählen Sie die Integrationsapplikation und danach Programm SI.

■ ∫f(x) SI

```
Set Vars; Select ∫var
X
```

Wählen Sie die Integrationsvariable X und geben Sie anschließend 0 als untere sowie 2 als obere Integrationsgrenze ein.

X
0 LLIM
2 ULIM

```
ULIM=2
LLIM ULIM ACC ∫
```


Da die Funktion

$$f(x) = \frac{\sin x}{x}$$

ein rein mathematischer Ausdruck ist, besteht die einzige Einschränkung für die Genauigkeit der Funktion in Rundungsfehlern durch den Rechner. Die Vorgabe eines Genauigkeitsfaktors von 0,00000000001 (1×10^{-11}) wäre analytisch gesehen also durchaus sinnvoll.

E 11 +/- ACC

```
ACC=0,00000000001
LLIM ULIM ACC ∫
```

Berechnen Sie das Integral.

∫

```
∫=1,6054129768
LLIM ULIM ACC ∫
```

Überprüfen Sie die Fehlerabschätzung für die Integration.

x≷y

```
x: 2,10542218026E-11
LLIM ULIM ACC ∫
```

Die Fehlerabschätzung für die Integration ist nur für die letzte Stelle des Integrals signifikant. Als Rechenzeit wurden etwa 19 Sekunden beansprucht. Wenn ein etwas ungenaueres Ergebnis in Kauf genommen werden kann, ist eine Verkürzung der Rechenzeit möglich. Versuchen Sie es z.B. mit einem Genauigkeitsfaktor von 0,001.

,001 ACC ∫

```
∫=1,60541531589
LLIM ULIM ACC ∫
```

Überprüfen Sie die Fehlerabschätzung für die Integration.

x≷y

```
x: 1,60600822892E-3
LLIM ULIM ACC ∫
```

Der angezeigte Fehler ist nun viel größer. Andererseits ist er noch immer klein im Vergleich mit dem Wert des Integrals, und als Rechenzeit wurden nur drei Sekunden beansprucht.

Verlassen Sie die Integrationsapplikation und stellen Sie wieder FIX 4 als Anzeigeformat ein.

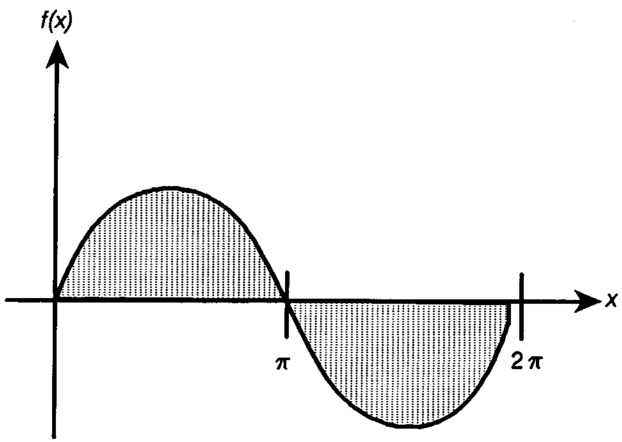
[EXIT] [EXIT] [EXIT]
■[DISP] FIX 4 [ENTER]

```
Y: 1,6054
x: 0,0016
```

Beispiel: Aufgabenstellung mit relativ großer Fehlerabschätzung für Integration. Im vorangehenden Beispiel war die Fehlerabschätzung relativ klein im Vergleich zum Wert des Integrals. Dies ist darauf zurückzuführen, daß der Funktionswert innerhalb des Integrationsintervalls immer positiv war. Betrachten Sie nun nachstehende einfache Funktion:

$$f(x) = \sin x$$

Integrieren Sie die Funktion von $x = 0$ bis $x = 6$ (Radiant).



Die Abbildung veranschaulicht, daß es sich um einen sehr kleinen Wert für das Integral handelt, da die Fläche im Intervall 0 bis π fast durch die Fläche im Intervall π bis 6 aufgehoben wird.

Schreiben Sie ein Programm, welches die Funktion ausdrückt.

```
00 { 14-Byte Prgm }
01 LBL "SIN"
02 MVAR "X"
03 RCL "X"
04 SIN
05 END
```

Stellen Sie RAD als Winkelmodus ein. Wählen Sie die Integrationsapplikation und anschließend das Programm SIN.

■[MODES] RAD
■[∫f(x)]
SIN

```
Set Vars; Select ∫var
X
```

Wählen Sie die Integrationsvariable X; geben Sie die untere und obere Integrationsgrenze (0 und 6) ein und spezifizieren Sie 0,01 als Genauigkeitsfaktor. Integrieren Sie anschließend über x.

X
0 LLIM
6 ULIM
,01 ACC
∫

```
∫=0,0398
LLIM ULIM ACC ∫
```

Überprüfen Sie die Fehlerabschätzung für die Integration.

[x≷y]

```
x: 0,0398
LLIM ULIM ACC ∫
```

Die Fehlerabschätzung fällt hier groß im Vergleich zum Wert des Integrals aus.

Verlassen Sie die Integrationsapplikation.

[EXIT] [EXIT] [EXIT]

```
Y: 0,0398
x: 0,0398
```

Mögliche Ursachen für unkorrekte Ergebnisse

Obwohl der Integrationsalgorithmus im HP-42S einer der besten verfügbaren Algorithmen ist, kann er Ihnen in bestimmten Situationen – wie fast alle Algorithmen für numerische Integration – ein unkorrektes Ergebnis liefern. *Die Wahrscheinlichkeit dafür ist jedoch sehr gering.* Der Algorithmus ist so ausgelegt, daß er für praktisch alle glatt verlaufenden Funktionen zuverlässige Ergebnisse liefert. Nur bei extrem sprunghaft verlaufenden Funktionen gehen Sie ein gewisses Risiko ein, ein ungenaues Ergebnis zu erhalten. Solche Funktionen kommen in physikalischen Problemstellungen jedoch kaum vor und können gegebenenfalls leicht erkannt und bearbeitet werden.

Beispiel: Bedingung, die zu ungenauem Ergebnis führt.
Betrachten Sie z.B. die Approximation von

$$\int_0^{\infty} xe^{-x}\,dx$$

Da Sie dieses Integral numerisch auswerten, ist es naheliegend (aber dennoch irreführend), die obere Integrationsgrenze mit einer relativ großen Zahl (z.B. 100 000) anzugeben. Versuchen Sie es und sehen Sie, was passiert. Schreiben Sie zuerst ein Programm, das $f(x)$ ausdrückt.

```
00 { 17-Byte Prgm }
01 LBL "XEX"
02 MVAR "X"
03 RCL "X"
04 ENTER
05 +/-
06 E↑X
07 ×
08 END
```

Wählen Sie die Integrationsapplikation und danach das Programm XEX.

■ ∫f(x) XEX

```
Set Vars; Select ∫var
X
```


Wählen Sie die Integrationsvariable X; geben Sie anschließend die untere und obere Integrationsgrenze (0 und 5) sowie einen Genauigkeitsfaktor von 0,001 ein.

X
0 LLIM
E 5 ULIM
,001 ACC

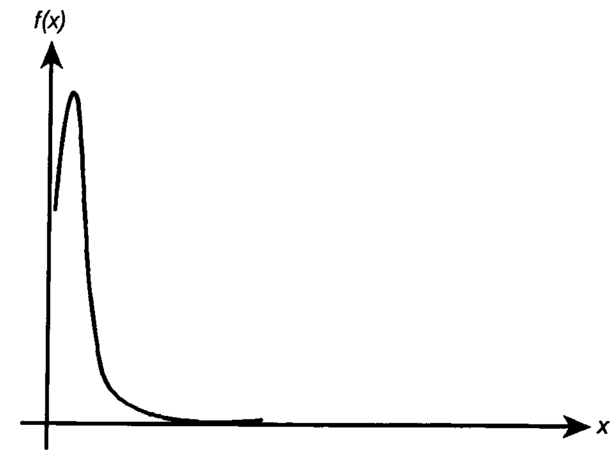
```
ACC=0,0010
LLIM ULIM ACC ∫
```

Integrieren Sie über x. (Verlassen Sie nicht die Integrationsapplikation; diese Funktion ist im nächsten Abschnitt erneut zu integrieren.)

∫

```
∫=0,0000
LLIM ULIM ACC ∫
```

Die vom Rechner ermittelte Lösung ist mit Sicherheit unkorrekt; das Integral von $f(x) = xe^{-x}$, von 0 bis ∞ hat exakt den Wert 1. Das Problem ist aber *nicht*, daß Sie ∞ durch 100 000 dargestellt haben, da das Integral dieser Funktion von 0 bis 100 000 fast den Wert 1 hat. Der Grund für das unkorrekte Ergebnis wird offensichtlich, wenn Sie den Graphen von $f(x)$ über das Integrationsintervall betrachten:



Der Graph hat einen Spike nahe des Ursprungs (hier stark vergrößert dargestellt). Da keine Stützstelle den Spike des Graphen entdeckte, nahm der Algorithmus an, daß die Funktion über das ganze Integrationsintervall gleich Null ist. Sogar bei erhöhter Anzahl von Stützstellen (durch einen Genauigkeitsfaktor von 1×10^{-11}) würde keine der zusätzlichen Stützstellen den Spike entdecken, wenn diese Funktion über das vorliegende Intervall integriert wird.

Aufteilen des Integrationsintervalls. Wenn Sie die Zuverlässigkeit des Ergebnisses bezweifeln, sollten Sie das Integrationsintervall in zwei oder mehrere Teilintervalle aufteilen. Integrieren Sie die Funktion über jedes Teilintervall und addieren Sie die Teilergebnisse. Daduch wird die Funktion über neue Stützstellen ausgewertet, wobei die Wahrscheinlichkeit, daß eventuell zuvor verborgene Spikes erfaßt werden, zunimmt. Ist die ursprüngliche Approximation korrekt, so entspricht sie der Summe der Approximationen über die Teilintervalle.

Beispiel: Aufteilen des Integrationsintervalls. Betrachten Sie nochmals das Integral

$$\int_0^{\infty} xe^{-x}\, dx$$

Approximieren Sie das Integral durch Aufteilen des ursprünglichen Intervalls in drei Teilintervalle: das erste von 0 bis 10, das zweite von 10 bis 100 und das dritte von 100 bis 100 000.

Zuerst ist das Intervall von 0 bis 10 zu integrieren. Wenn Sie sich noch in der Integrationsapplikation befinden, geben Sie einfach den neuen Wert für *ULIM* ein.

10 ULIM
∫

```
∫=0,9995
LLIM ULIM ACC  ∫
```

Das Ergebnis ist nahe bei 1. Integrieren Sie nun das Intervall von 10 bis 100.

10 LLIM
100 ULIM
∫

```
∫=0,0005
LLIM ULIM ACC  ∫
```

Das Ergebnis ist nahe bei 0. Die Summe der Approximationen über die zwei Teilintervalle ist 1. Integrieren Sie schließlich von 100 bis 100 000. (Bleiben Sie in der Integrationsapplikation, da diese Funktion im nächsten Abschnitt wieder zu integrieren ist.)

100 LLIM
100000 ULIM
∫

```
∫=0,0000
LLIM ULIM ACC  ∫
```


Das Integral über das 3. Teilintervall entspricht Null. Als Gesamtsumme der Teilintervalle ergibt sich somit 1.

Bedingungen für verlängerte Rechenzeiten

Im ersten Beispiel des vorangehenden Abschnitts lieferte der Algorithmus ein unkorrektes Ergebnis, weil er den Spike der Funktion $f(x) = xe^{-x}$ gar nicht entdeckte. Dies geschah deshalb, weil die Variation der Funktion im Vergleich zur Intervallbreite zu schnell war. Im zweiten Beispiel erhielten Sie eine sehr gute Approximation durch Aufteilung des Integrationsintervalls in drei Teilintervalle. Für diese Funktion gibt es jedoch einen Bereich von Intervallen, die klein genug sind, um zu einem korrekten Ergebnis zu führen; allerdings auf Kosten einer verlängerten Rechenzeit.

Beispiel: Approximation für Obergrenze, was zu verlängerter Rechenzeit führt. Betrachten Sie erneut das Integral

$$\int_0^{\infty} xe^{-x}\, dx$$

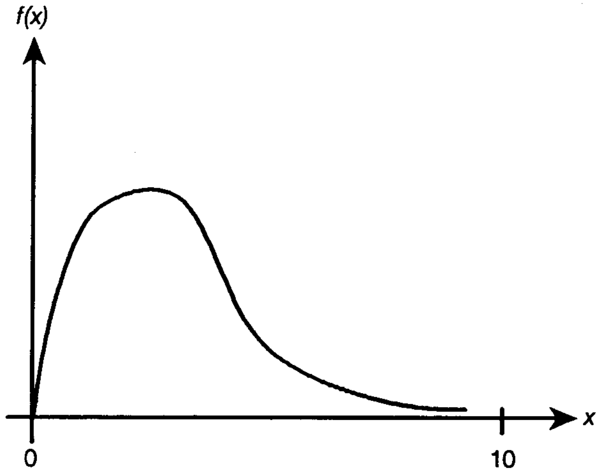
Approximieren Sie das Integral durch Integration über das Intervall (0, 1000).

Geben Sie die neuen Werte für *LLIM* und *ULIM* ein. Integrieren Sie anschließend über x.

0 LLIM
1000 ULIM
∫

```
∫=1,0000
LLIM ULIM ACC  ∫
```

Dies ist das richtige Ergebnis, aber die Berechnung dauert sehr lange. Um dies zu verstehen, vergleichen Sie den Graphen der Funktion zwischen $x = 10$ und $x = 10^3$ (welcher dem auf Seite 141 abgebildeten sehr ähnelt), mit dem folgenden Graphen der Funktion zwischen $x = 0$ und $x = 10$.



Sie können erkennen, daß die Funktion nur für sehr kleine Werte von x "interessant" ist. Für größere Werte von x ist die Funktion uninteressant, da sie stetig und glatt in einer vorhersehbaren Weise abfällt.

Der Algorithmus erhöht die Dichte der Stützpunkte, bis genügend Daten über die Funktion vorliegen, um eine Approximation zu berechnen, die sich bei zunehmender Stützstellenzahl nur unwesentlich ändert. Im vorherigen Abschnitt, bei der Auswertung des Integrals zwischen 0 und 10, untersuchte der Algorithmus die Funktion nur bei Werten, die interessant sind und in einem glatten Bereich liegen. Nach den ersten Iterationen trugen die weiteren Stützstellen zu keinen neuen Erkenntnissen über das Verhalten der Funktion bei und der Integrationsprozeß wurde abgebrochen.

Im letzten Beispiel bezogen sich die meisten Stützstellen auf einen Funktionsbereich, wo sich die Steigung kaum veränderte. Der Algorithmus erkennt, daß die wenigen Stützstellen für sehr kleine Werte von x zu Funktionswerten führen, die sich zwischen den Iterationen deutlich ändern. Demzufolge muß die Funktion an zusätzlichen Stützstellen ausgewertet werden, bevor die Ungleichheit zwischen aufeinanderfolgenden Approximationen ausreichend klein wird.

Für eine Approximation des Integrals mit der gleichen Genauigkeit über das große sowie über das kleine Intervall muß die Dichte der Stützstellen im interessanten Funktionsbereich gleich sein. Um die gleiche Stützstellendichte zu erhalten, ist die gesamte Anzahl der erforderlichen Stützstellen über das große Intervall viel größer als die erforderliche Anzahl für das kleine Intervall. Folglich sind im größeren Intervall zusätzliche Iterationen erforderlich, um eine Approximation gleicher Genauigkeit zu erreichen, und deshalb wird für diese Berechnung beträchtlich mehr Zeit benötigt.

5

Matrizen

Dieses Kapitel baut auf den Informationen auf, welche in Kapitel 14 des Benutzerhandbuchs enthalten sind. Es werden folgende Themen behandelt:

- Verwenden des Matrix-Editors und der Indizierungsfunktionen.
- Vektorrechnung.
- Lösen linearer Gleichungssysteme.
- Verwenden des Lösers für lineare Gleichungssysteme.
- Matrixoperationen in Programmen.

Verwenden des Matrix-Editors und der Indizierungsfunktionen

Im nachstehenden Beispiel sind folgende Operationen auszuführen:

- Erzeugen einer Matrix.
- Verwenden des Matrix-Editors zum Modifizieren von Daten.
- Interaktives Verwenden der Indizierungs- und Statistikfunktionen.

Beispiel: Akkumulieren von meteorologischen Daten. Dr. Brausewind, ein bekannter Meteorologist, hat folgende Wetterdaten notiert und möchte diese in Form einer Matrix im HP-42S speichern.

Tag	Temp	Wind	Feucht
1	17	8	54
2	19	14	36
3	24	4	72

Erzeugen einer benannten Matrix

Erzeugen Sie eine 4×4 Matrix "*WETT*".

4 [ENTER] ■[MATRIX] [▼] DIM
[ENTER] WETT [ENTER]

```
x: 4,0000
DOT CROSS UVEC DIM INDEX EDITN
```

Verwenden des Matrix-Editors

Rufen Sie den Matrix-Editor auf und wählen Sie die zuvor erzeugte Matrix.

EDITN WETT

```
1:1=0,0000
← OLD ↑ ↓ GOTO →
```

Belegen Sie Element 1:1 mit dem Alpha-String TAG. (Denken Sie daran, daß zur Ausführung von [ASTO] die Taste [STO] im ALPHA Modus zu drücken ist.)

■[ALPHA] TAG [ASTO] [.]
ST X [EXIT]

```
1:1="TAG"
← OLD ↑ ↓ GOTO →
```

Füllen Sie die restlichen Elemente in Zeile 1 mit den korrespondierenden Alpha-Strings der Tabelle auf. (Die Tastenfolge für Element (1:2) finden Sie nachstehend.)

→ ■[ALPHA] TEMP [ASTO]
[.] ST X [EXIT] ...

```
1:4="FEUCHT"
← OLD ↑ ↓ GOTO →
```

Füllen Sie nun die restlichen Elemente mit den zugehörigen Daten auf.

```
    →  1  →          4:4=72_
17  →  8              ←  OLD  ↑  ↓  GOTO  →
    →  54 →
2   →  19
    →  14 →
36  →  3
    →  24 →
4   →  72
```

Dr. Brausewind bemerkt, daß sein Assistent die Temperatur für Tag 1 falsch aufgezeichnet hat: es waren 27 °C, nicht 17.

```
GOTO 2 [ENTER] 2 [ENTER] 27 [EXIT]      x: 27,0000
                                        DOT CROSS UVEC DIM INDEX EDITN
```

Einige Tage später sind Daten für Tag 4 hinzuzufügen: Temperatur = 27, Windgeschwindigkeit = 5 und Luftfeuchtigkeit = 76. Stellen Sie zuerst den Rechner auf Zuwachs-Modus ein, um eine neue Zeile für die Matrix zu erzeugen.

```
EDITN WETT  ←         5:1=0,0000
[▼] GROW [▲]          ←  OLD  ↑  ↓  GOTO  →
→
```

Geben Sie die neuen Daten ein.

```
4  →  27              5:4=76_
   →  5  →  76        ←  OLD  ↑  ↓  GOTO  →
```

Unmittelbar danach bemerkt Dr. Brausewind, daß er die Daten für Tag 5 anstatt für Tag 4 eingegeben hat. Für Tag 4 wurden folgende Werte aufgezeichnet: Temperatur = 18, Windgeschwindigkeit = 12, Luftfeuchtigkeit = 41. Ändern Sie zuerst den Wert in Element 5:1 auf 5.

```
←  ←                  5:1=5_
←  5                  ←  OLD  ↑  ↓  GOTO  →
```

Geben Sie nun die neue Zeile ein.

```
[▼] INSR              5:1=0,0000
                      INSR  DELR  WRAP GRO■
```


Geben Sie die tatsächlichen Werte für Tag 4 ein.

```
[▲] 4  →  18          5:4=41_
→  12  →  41          ←  OLD  ↑  ↓  GOTO  →
```

Verlassen Sie die Matrix-Applikation.

```
[EXIT] [EXIT]         Y: 4,0000
                      X: 41,0000
```

Interaktive Verwendung von Indizierungs- und Statistikfunktionen

Dr. Brausewind möchte nun statistische Auswertungen von Segmenten seiner gespeicherten Wetterdaten vornehmen. Er möchte die mittlere Temperatur sowie die Windgeschwindigkeit während der 5 Tage berechnen. Er führt dazu GETM aus, um im X-Register eine 5×2 Untermatrix zu erzeugen, welche die Temperatur- und Windgeschwindigkeitswerte enthält. Anschließend führt er $\Sigma+$ aus, um die Untermatrix-Daten in den Summationsregistern (Statistikregister) zu speichern; durch Wählen des STAT Menüs läßt sich dann auf einfache Weise der Mittelwert berechnen. (Denken Sie daran, daß die $\Sigma+$ Funktion *automatisch* die Daten einer n-Zeilen $\times$ 2-Spalten-Matrix in die momentan definierten Summationsregister kopiert. Weitere Informationen hierzu finden Sie in Kapitel 15 des Benutzerhandbuchs.)

Spezifizieren Sie *WETT* als indizierte Matrix.

```
■[MATRIX] [▼] INDEX WETT      x: 41,0000
                              DOT CROSS UVEC DIM INDEX EDITN
```

Stellen Sie die Indexzeiger auf Element 2:2 (der erste Temperaturwert).

```
2 [ENTER] [▼] STOIJ           x: 2,0000
                              STOIJ RCLIJ STOEL RCLEL PUTM GETM
```

Rufen Sie nun die 5×2 Untermatrix ab, die die Werte für Temperatur und Windgeschwindigkeit enthält.

```
5 [ENTER] 2 GETM              x: [ 5x2 Matrix ]
                              STOIJ RCLIJ STOEL RCLEL PUTM GETM
```

Löschen Sie die Summationsregister und kopieren Sie danach die Untermatrix-Daten in die Summationsregister. (Falls die Meldung `Nonexistent` angezeigt wird, ist die momentane SIZE Zuordnung zu klein.)

■[CLEAR] CLΣ
■[TOP.FCN] Σ+

```
x: 5,0000
STOIJ RCLIJ STOEL RCLEL PUTM GETM
```

Rufen Sie das STAT Menü auf und berechnen Sie die Durchschnittstemperatur.

■[STAT] MEAN

```
x: 23,0000
Σ+ SUM MEAN WMN SDEV CFIT
```

Berechnen Sie die durchschnittliche Windgeschwindigkeit.

[x≷y]

```
x: 8,6000
Σ+ SUM MEAN WMN SDEV CFIT
```

Als Mittelwert für Temperatur und Windgeschwindigkeit erhalten Sie die Werte 23 bzw. 8,6.

Verlassen Sie das STAT Menü.

[EXIT]

```
y: 22,4000
x: 8,6000
```

Matrix-Dienstprogramme

Die nachstehenden Routinen verwenden bestehende Matrixfunktionen zum Erzeugen hilfreicher Matrix-Dienstprogramme.

Berechnen der Spaltensumme einer Matrix. SSUM berechnet die Spaltensumme der Matrix im X-Register. (Die Spaltensumme einer Matrix A ist eine einzeilige Matrix, deren Elemente der Spaltensumme der korrespondierenden Matrix A entsprechen.) Die resultierende Matrix wird in das X-Register zurückgegeben.

```
00 { 14-Byte Prgm }
01 LBL "SSUM"
02 TRANS
03 RSUM
```



```
04 TRANS
05 END
```

Berechnen der Spaltennorm einer Matrix. SNRM berechnet die Spaltennorm der Matrix im X-Register. (Die Spaltennorm einer Matrix A ist der größte Wert (über alle Spalten) von den Summen der Absolutbeträge aller Spaltenelemente.) Das Ergebnis wird in das X-Register zurückgegeben.

```
00 { 12-Byte Prgm }
01 LBL "SNRM"
02 TRANS
03 RNRM
04 END
```

Konjugieren einer komplexen Matrix. Um eine komplexe Matrix zu konjugieren:

1. Bringen Sie die Matrix in das X-Register.
2. Drücken Sie ■[COMPLEX].
3. Drücken Sie [+/-].
4. Drücken Sie ■[COMPLEX].

Die konjugiert komplexe Matrix wird in das X-Register zurückgegeben.

Berechnen der Matrix-Summe. MSUM berechnet die Matrix-Summe (Summe aller Elemente) der Matrix im X-Register. Das Ergebnis wird in das X-Register zurückgegeben.

```
00 { 18-Byte Prgm }
01 LBL "MSUM"
02 XEQ "SSUM"
03 RSUM
04 DET
05 END
```

Auffinden des größten und kleinsten Elements einer Matrix. MINMAX ermittelt das größte oder kleinste Element der *reellen* Matrix im X-Register. Das gefundene Element wird in das X-Register zurückgegeben. Die Indizes des Elements werden im Y- und Z-Register gespeichert (Spaltennummer in Y, Zeilennummer in Z). Setzen Sie Flag 09, um das größte Element zu ermitteln; löschen Sie Flag 09 zum Auffinden des kleinsten Elements.

Programm: / **Kommentar:**

```
00 { 61-Byte Prgm }
01 LBL "MINMAX"
```

```
02 STO "MINMAX"
03 INDEX "MINMAX"
04 RCLEL
05 GTO 03
```

Zeile 02 – 05: Speichere momentane Matrix im X-Register in *MINMAX*, indiziere *MINMAX* und lege Element 1:1 als momentanes Maximum- oder Minimum-Element fest.

```
06 LBL 01
07 RCLEL
08 FS? 09
09 GTO 02
10 X≥Y?
11 GTO 04
12 GTO 03
```

Zeile 06 – 12: Ist Flag 09 gelöscht, überprüfe, ob momentanes Element größer als momentanes Minimum ist. Falls ja, springe zu Label 04 (momentanes Minimum beibehalten), ansonsten springe zu Label 03 (speichern des momentanen Elements als neues Minimum.)

```
13 LBL 02
14 X≤Y?
15 GTO 04
```

Zeile 13 – 15: Ist Flag 09 gesetzt, überprüfe, ob momentanes Element kleiner als momentanes Maximum ist. Falls ja, springe zu Label 04 (momentanes Maximum beibehalten), ansonsten mache momentanes Element zum neuen Maximum.

```
16 LBL 03
17 RCLIJ
18 RCL ST Z
19 ENTER
```

Zeile 16 – 19: Mache momentanes Element zu neuem Maximum oder Minimum.


```
20 LBL 04
21 R↓
22 J+
23 FC? 77
24 GTO 01
```

Zeile 20 – 24: Behalte momentanes Maximum bzw. Minimum bei.

```
25 END
```

Sortieren einer Matrix. SORT sortiert die Zeilen der Matrix im X-Register in aufsteigender Reihenfolge, entsprechend der Werte in Spalte 1. Die sortierte Matrix wird in das X-Register zurückgegeben.

Programm: / **Kommentar:**

```
00 { 81-Byte Prgm }
01 LBL "SORT"
```

```
02 STO "SORTMAT"
03 INDEX "SORTMAT"
```

```
04 LBL 01
05 I+
06 FS? 76
07 GTO 04
08 RCLIJ
09 X<>Y
10 RCLEL
```

Zeile 07 – 10: Lege die zu sortierende Zeilennummer fest. (Im ersten Durchlauf ist Zeile 2 gegen Zeile 1 zu sortieren. Im zweiten Durchlauf ist Zeile 3 gegen Zeile 1 und 2 zu sortieren, usw.) Fortsetzung, bis alle Zeilen sortiert sind.

```
11 LBL 02
12 I-
13 RCLEL
14 FS? 76
15 GTO 03
16 X≤Y?
17 GTO 03
18 R↓
19 RCLIJ
20 RCL+ ST Y
21 R<>R
22 R↓
23 R↓
```

Zeile 11 – 24: Verschiebe nacheinander die "Sortierzeile" aufwärts in der Matrix, bis der Wert in Spalte 1 größer als der vorherige Wert in Spalte 1 ist.

```
24 GTO 02

25 LBL 03
26 R↓
27 R↓
28 1
29 STOIJ
30 GTO 01
31 LBL 04
32 RCL "SORTMAT"

33 END
```

Zeile 25–32: Erhöhe die Nummer der "Sortierzeile". Führt die Erhöhung zu einem Umsprung des Indexzeigers, zeige die sortierte Matrix in X an und beende das Programm.

Vektorrechnung

Vektoren stellen eine spezielle Untermenge von Matrizen dar. Sie können einen Vektor entweder über eine $1 \times n$-Spalten-Matrix oder eine n-Zeilen $\times$ 1-Spalte-Matrix darstellen.

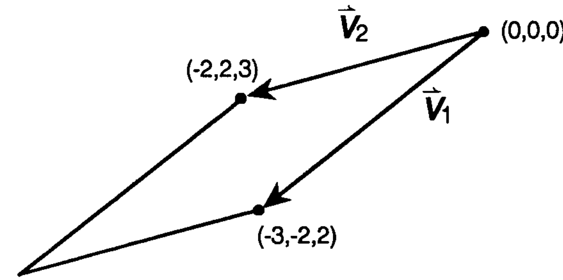
Geometrie

Die Fläche eines Parallelogramms kann durch folgende Gleichung berechnet werden:

$$A = \text{Frobenius Norm (Betrag) von } (V\mathbf{V}_1 \times \mathbf{V}_2)$$

wobei $(\mathbf{V}_1 \times \mathbf{V}_2)$ das Kreuzprodukt der Vektoren $\mathbf{V}_1$ and $\mathbf{V}_2$ darstellt.

Beispiel: Fläche eines Parallelogramms. Berechnen Sie die Fläche des folgenden Parallelogramms:



Erzeugen Sie die Vektoren $\mathbf{V}_1$ und $\mathbf{V}_2$.

■[MATRIX]
1 [ENTER] 3 [▼] DIM
[ENTER] V1 [ENTER]
1 [ENTER] 3 DIM
[ENTER] V2 [ENTER]

x: 3,0000
DOT CROSS UVEC DIM INDEX EDITN

Geben Sie die Werte für jedes Element in $\mathbf{V}_1$ ein.

EDITN V1
3 [+/-] →
2 [+/-] →
2 [EXIT]

x: 2,0000
DOT CROSS UVEC DIM INDEX EDITN

Geben Sie die Werte für jedes Element in $\mathbf{V}_2$ ein.

EDITN V2
2 [+/-] →
2 →
3 [EXIT]

x: 3,0000
DOT CROSS UVEC DIM INDEX EDITN

Berechnen Sie die Fläche.

[RCL] V1 [RCL] V2
CROSS ■[CATALOG] FCN
[▼] ... FNRM

x: 15,0000
DOT CROSS UVEC DIM INDEX EDITN

Die Fläche des Parallelogramms ist 15,0000.

Verlassen Sie die MATRIX Applikation.

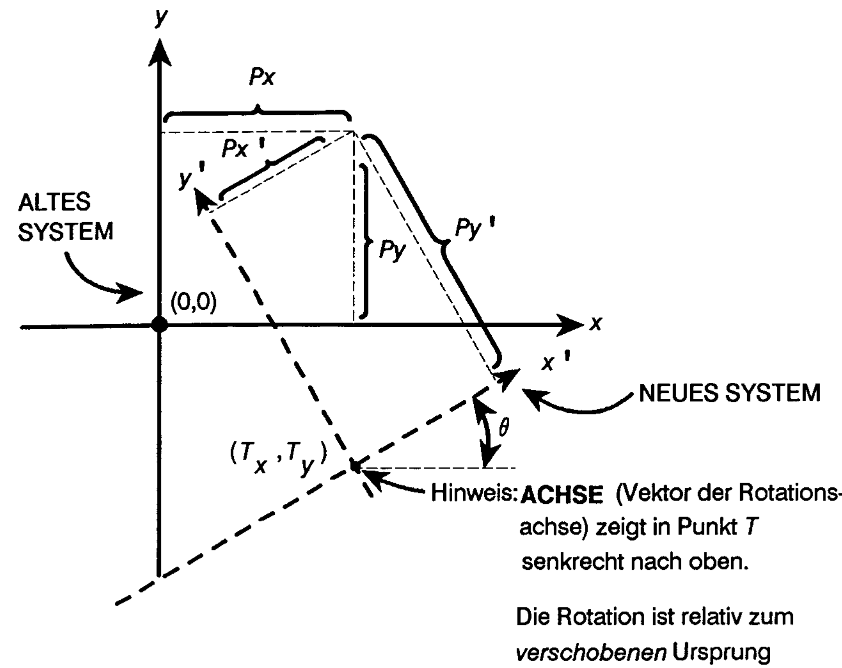
[EXIT]

```
Y: 3,0000
X: 15,0000
```

Koordinatentransformationen

Bei vielen Aufgabenstellungen der Kinematik oder Mechanik ist es erforderlich, Koordinaten von einem System in ein anderes System überzuführen. Für Koordinatentransformationen sind folgende Operationen erforderlich:

- Berechnen des Einheitsvektors.
- Addieren von Vektoren.
- Berechnen des Skalarprodukts zweier Vektoren.
- Multiplizieren von Vektoren.
- Berechnen des Kreuzprodukts zweier Vektoren.



Die Gleichung zur Koordinatentransformation eines Punktes aus dem alten System in das neue System lautet:

$$\mathbf{P}' = [(\mathbf{P} - \mathbf{T})\cdot\mathbf{n}]\mathbf{n}(1 - \cos\theta) + (\mathbf{P} - \mathbf{T})\cos\theta + [(\mathbf{P} - \mathbf{T})\times\mathbf{n}]\sin\theta$$

Die Gleichung zur Koordinatentransformation eines Punktes aus dem neuen System in das alte System lautet:

$$\mathbf{P} = [(\mathbf{P}'\cdot\mathbf{n})\,\mathbf{n}\,(1 - \cos\theta) + \mathbf{P}'\cos\theta + (\mathbf{P}'\times\mathbf{n})\sin(-\theta)] + \mathbf{T}$$

wobei:

$\mathbf{P}'$ = Koordinaten des Punktes im neuen System.
$\mathbf{P}$ = Koordinaten des Punktes im alten System.
$\mathbf{T}$ = Ursprung des neuen Systems.
$\mathbf{n}$ = Einheitsvektor der Achse, über welche Rotation ausgeführt wird.
θ = Rotationswinkel.

Beachten Sie, daß die Translation vor der Rotation erfolgt. Die Rotation ist relativ zum verschobenen Ursprung.

Das nachstehende Programm KOORD erlaubt Ihnen das Auffüllen der Vektoren **P**, (oder **P´**), **T** und **ACHSE** mit Daten, indem über das Programm der Matrix-Editor aufgerufen wird. Weiterhin können Sie entweder eine Alt-Nach-Neu oder eine Neu-Nach-Alt Transformation spezifizieren. (**ACHSE** ist der Vektor für die Rotationsachse. KOORD speichert die für **ACHSE** eingegebenen Werte in der Variablen *n* und berechnet anschließend den Einheitsvektor **n**.)

Um KOORD einzutippen: Erzeugen Sie zuerst die Variablen *P*, *T*, *P´*, *n* und ∡, bevor Sie mit der Programmeingabe beginnen.

Hier eine kommentierte Liste von KOORD.

Programm: / **Kommentar:**

```
00 { 217-Byte Prgm }
01 LBL "KOORD"
```

```
02 EXITALL
03 CLMENU
04 "P"
05 KEY 1 GTO 01
06 "T"
07 KEY 2 XEQ 02
08 "ACHSE"
09 KEY 3 XEQ 03
10 "∡"
11 KEY 4 XEQ 04
```

Zeile 02–11: Erzeuge Hauptmenü.

```
12 LBL 98
13 MENU
14 STOP
15 GTO 98
```

Zeile 12–15: Zeige Hauptmenü an.

```
16 LBL 01
17 "P"
18 XEQ 99
19 "N→A"
20 KEY 5 GTO 05
21 "A→N"
22 KEY 6 GTO 06
```

Zeile 16–22: Zeige das Untermenü zum Edieren von Vektor **P** (oder **P´**) an und wähle die Richtung der Transformation.

```
23 LBL 97
24 MENU
25 CF 00
26 STOP
27 GTO 97
```

Zeile 23–27: Zeige Untermenü an.

```
28 LBL 02
29 "T"
30 GTO 99
31 LBL 03
32 "n"
```

Zeile 28–32: Bringe die Vektornamen **T** und **n** in das Alpha-Register, um die Vektoren zu erzeugen.

```
33 LBL 99
34 CLMENU
35 ASTO ST L
36 1
37 ENTER
38 3
39 DIM IND ST L
40 EDITN IND ST L
41 "←"
42 KEY 1 XEQ 11
43 "→"
44 KEY 2 XEQ 12
45 KEY 9 GTO "KOORD"
46 RTN
```

Zeile 33–46: Erzeuge einen 1 × 3 Vektor **P**, **T**, oder **n** und erlaube dessen Modifikation über den Matrix-Editor. Erzeuge Menüfelder für Matrix-Editor und fordere zur Dateneingabe auf.

```
47 LBL 11
48 ←
49 RTN
50 LBL 12
51 →
52 RTN
```

Zeile 47–52: Führe Funktionen des Matrix-Editors aus.

```
53 LBL 04
54 INPUT "∡"
55 RTN
```

Zeile 53–55: Eingabeaufforderung für Wert von ∡.

```
56 LBL 05
57 SF 00
```

Zeile 56–57: Setze Flag 00 für Neu-Nach-Alt Transformation.

```
58 LBL 06
59 EXITALL
60 RCL "P"
61 FC? 00
62 RCL- "T"
63 STO "P'"
64 RCL "n"
65 UVEC
66 STO "n"
67 DOT
68 1
69 RCL "∡"
70 COS
71 -
72 RCL× "n"
73 ×
74 RCL "∡"
75 COS
76 RCL× "P'"
77 +
78 RCL "P'"
79 RCL "n"
80 CROSS
81 RCL "∡"
82 FS? 00
83 +/-
84 SIN
85 ×
86 +
87 FS? 00
88 RCL+ "T"
89 STO "P"
90 GTO 01
```

Zeile 58–90: Werte die Transformationsgleichung aus. Ist Flag 00 gelöscht, berechne Alt-Nach-Neu Transformation; ist Flag 00 gesetzt, berechne Neu-Nach-Alt Transformation.

```
91 END
```


Anwenden von KOORD:

1. Drücken Sie [XEQ] KOOR .
2. Drücken Sie T und geben Sie danach die Werte für **T** ein, indem Sie das Menü des Matrix-Editors verwenden. Drücken Sie [EXIT] zur Rückkehr zum Hauptmenü.
3. Drücken Sie ACHSE und geben Sie danach die Werte für die Rotationsachse ein, wobei wieder das Menü des Matrix-Editors zu benutzen ist. Drücken Sie [EXIT] zur Rückkehr zum Hauptmenü. Beachten Sie, daß KOORD die Rotationsachse in Variable *n* speichert, dann deren Einheitsvektor berechnet und diesen zurück in *n* speichert. Wenn Sie nach der Ausführung einer dreidimensionalen Transformation ACHSE drücken, sehen Sie die neuberechneten Elemente des Einheitsvektors, nicht die ursprüngliche Rotationsachse.

   Für eine zweidimensionale Transformation ist (0, 0, 1) als Rotationsachse vorzugeben.
4. Drücken Sie ∡ , geben Sie danach einen Wert für ∡ ein und drücken Sie [R/S].
5. Drücken Sie P und geben Sie dann die Werte für die Elemente von **P** (oder **P′**) ein, indem Sie das Menü des Matrix-Editors verwenden. Drücken Sie danach A→N , um eine Alt-Nach-Neu Transformation auszuführen; drücken Sie N→A für eine Transformation aus dem neuen in das alte System. Die Berechnungen werden nun ausgeführt.

Beispiel: Dreidimensionale Translation mit Rotation. Ein dreidimensionales Koordinatensystem wird von (0, 0, 0) nach (2,45; 4,00; 4,25) verschoben. Nach der Translation erfolgt eine Rotation um 62,5° über die (0, −1, −1) Achse. Berechnen Sie die neuen Koordinaten eines Punktes mit den Koordinaten (3,90; 2,10; 7,00) im alten System.

Für diese Aufgabenstellung gilt:

P = (3,90; 2,10; 7,00)
T = (2,45; 4,00; 4,25)
ACHSE = (0, −1, −1)
∡ = 62,5°

Stellen Sie als Anzeigeformat FIX 2 ein und spezifizieren Sie Grad als Winkelmodus. Führen Sie danach das Programm KOORD aus.

■[DISP] FIX 02
■[MODES] DEG
[XEQ] KOOR

```
x: 0,00
P | T | ACHSE | ∡ | | 
```

Geben Sie die Elemente von T ein.

T
2,45 →
4 →
4,25 →
[EXIT]

```
x: 2,45
P | T | ACHSE | ∡ | | 
```

Geben Sie die Elemente der Rotationsachse ein.

ACHSE
→
1 [+/-] →
1 [+/-] →
[EXIT]

```
x: 0,00
P | T | ACHSE | ∡ | | 
```

Speichern Sie den Wert für ∡.

∡
62,5 [R/S]

```
x: 62,50
P | T | ACHSE | ∡ | | 
```

Geben Sie die Elemente von **P** ein.

P
3,9 →
2,1 →
7 →

```
1:1=3,90
← | → | | | N→A | A→N
```

Berechnen Sie die Transformation.

A→N

```
1:1=3,59
← | → | | | N→A | A→N
```

Element 1:1 von **P´** ist 3,59. Überprüfen Sie Element 1:2.

→

```
1:2=0,26
← | → | | | N→A | A→N
```


Überprüfen Sie Element 1:3.

→

```
1:3=0,59
← | → | | | N→A | A→N
```

Die Koordinaten des Punktes im neuen System lauten (3,59; 0,26; 0,59). Verlassen Sie das Programm KOORD und stellen Sie wieder FIX 4 als Anzeigeformat ein.

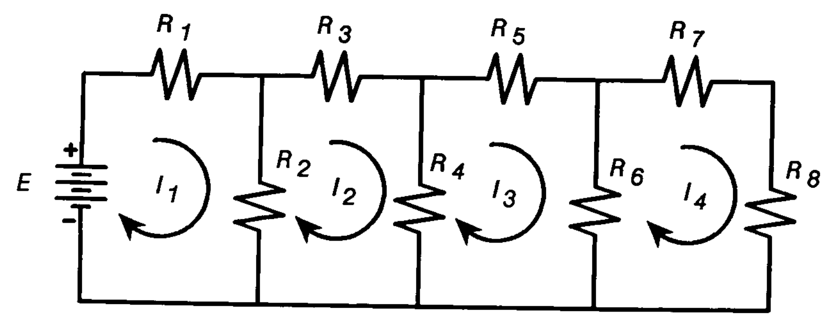
[EXIT] [EXIT]
■[DISP] FIX 4 [ENTER]

```
Y: 1,0000
x: 0,5891
```

Lösen linearer Gleichungssysteme

Die Berechnung der einzelnen Maschen in einem elektronischen Netzwerk erfolgt über das Auswerten einer Reihe von Gleichungen, welche in einem linearen Gleichungssystem zusammengefaßt sind. Die Anzahl der Gleichungen entspricht der Anzahl von Maschen des Netzwerks. Das erste Beispiel in diesem Abschnitt berechnet die Maschenströme in einem viermaschigen ohmschen Netzwerk (die Terme des linearen Gleichungssystems sind ausschließlich reellwertig). Das zweite Beispiel berechnet die Maschenströme in einem viermaschigen Netzwerk mit kapazitiven Anteilen (die Terme dieses linearen Gleichungssystems bestehen aus komplexen Zahlen).

Beispiel: Lösen eines reellwertigen linearen Gleichungssystems. Betrachten Sie nachstehendes ohmsches Netzwerk.



Wenden Sie die Maschenregel an, um die Maschenströme I_1, I_2, I_3, I_4 zu berechnen.

Bei den zu lösenden Gleichungen (über Variablen ausgedrückt) handelt es sich um:

1. $(R_1 + R_2)(I_1) - (R_2)(I_2) = E$
2. $-(R_2)(I_1) + (R_2 + R_3 + R_4)(I_2) - (R_4)(I_3) = 0$
3. $-(R_4)(I_2) + (R_4 + R_5 + R_6)(I_3) - (R_6)(I_4) = 0$
4. $-(R_6)(I_3) + (R_6 + R_7 + R_8)(I_4) = 0$

Bringen Sie die Gleichungen in Matrixform, wobei Sie die nachstehenden Werte für die Variablen einsetzen: $E = 34$ V und R_1 bis $R_8 = 1\,\Omega$.

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} 34 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Wählen Sie die Applikation zum Lösen linearer Gleichungssysteme (*SIMultaneous eQuations*) und spezifizieren Sie die Anzahl der Unbekannten.

■ [MATRIX] SIMQ 4 [ENTER]

```
x: 0,0000
MATA MATB MATX
```

Geben Sie die Werte für die Elemente der Koeffizientenmatrix *MATA* ein. (Die Tastenfolge zum Eingeben der ersten Zeile ist nachstehend erläutert.) Kehren Sie nach Eingabe aller Daten zum Hauptmenü zurück.

MATA 2 →
1 [+/-] →
0 →
0 → ...

[EXIT]

```
x: 3,0000
MATA MATB MATX
```

Speichern Sie die Werte für die Konstantenmatrix *MATB*.

MATB
34 ↓
0 ↓
0 ↓
0 [EXIT]

```
x: 0,0000
MATA MATB MATX
```


Berechnen Sie die Unbekannten.

MATX

```
1:1=21,0000
← OLD ↑ ↓ GOTO →
```

I_1 ist 21 A. Überprüfen Sie nun I_2.

↓

```
2:1=8,0000
← OLD ↑ ↓ GOTO →
```

Überprüfen Sie I_3.

↓

```
3:1=3,0000
← OLD ↑ ↓ GOTO →
```

Überprüfen Sie I_4.

↓

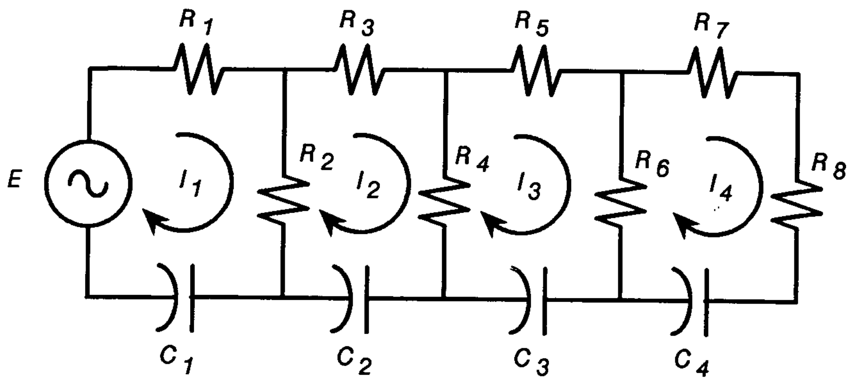
```
4:1=1,0000
← OLD ↑ ↓ GOTO →
```

Verlassen Sie den Matrix-Editor (bleiben Sie jedoch in der Applikation zum Lösen linearer Gleichungssysteme).

[EXIT]

```
x: 1,0000
MATA MATB MATX
```

Beispiel: Lösen eines linearen Gleichungssystems mit komplexen Termen. Betrachten Sie nun das folgende Netzwerk.



Der Kondensator in jeder Masche des Netzwerks bewirkt die Einführung eines komplexen Terms in jeder Maschengleichung:

1. $\left[(R_1 + R_2) - i\left(\frac{1}{\omega C_1}\right)\right](I_1) - (R_2)(I_2) = E$

2. $-(R_2)(I_1) + \left[R_2 + R_3 + R_4 - i\left(\frac{1}{\omega C_2}\right)\right](I_2) - (R_4)(I_3) = 0$

3. $-(R_4)(I_2) + \left[R_4 + R_5 + R_6 - i\left(\frac{1}{\omega C_3}\right)\right](I_3) - (R_6)(I_4) = 0$

4. $-(R_6)(I_3) + \left[R_6 + R_7 + R_8 - i\left(\frac{1}{\omega C_4}\right)\right](I_4) = 0$

Bringen Sie die Gleichungen in Matrixform, wobei Sie die folgenden Werte für die Variablen einsetzen: $E = 34$ V, R_1 bis $R_8 = 5\,\Omega$, $\omega = 100$ 1/s und C_1 bis $C_4 = 1$ F.

$$\begin{bmatrix} 10 - i0{,}01 & -5 & 0 & 0 \\ -5 & 15 - i0{,}01 & -5 & 0 \\ 0 & -5 & 15 - i0{,}01 & -5 \\ 0 & 0 & -5 & 15 - i0{,}01 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} 34 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Stellen Sie Rechtecksnotation für den Koordinatenmodus ein; definieren Sie *MATA* als komplexe Matrix.

■ [MODES] RECT
0 [ENTER] ■ [COMPLEX]
[STO] [x] MATA

```
x: 0,0000 i0,0000
MATA MATB MATX
```

Geben Sie die Werte für die Matrix ein. (Die Tastenfolge zum Eingeben der ersten Zeile ist nachstehend dargestellt.) Kehren Sie nach Eingabe aller Daten zum Hauptmenü zurück.

MATA
10 [ENTER] ,01 [+/-]
■ [COMPLEX] →
5 [+/-] →
0 →
0 → ...

[EXIT]

```
x: 15,0000 -i0,0100
MATA MATB MATX
```

Berechnen Sie *MATX*. (*MATB* hat den gleichen Inhalt wie im vorherigen Beispiel.)

MATX

```
1:1=4,2000 i0,0061
← OLD ↑ ↓ GOTO →
```

I_1 ist 4,2000 + i0,0061 A. Überprüfen Sie nun I_2.

↓

```
2:1=1,6000 i0,0037
← OLD ↑ ↓ GOTO →
```

Überprüfen Sie I_3.

[↓]

```
3: 1=0,6000 i0,0019
 ←  OLD  ↑  ↓  GOTO  →
```

Überprüfen Sie I_4.

[↓]

```
4: 1=0,2000 i0,0008
 ←  OLD  ↑  ↓  GOTO  →
```

Verlassen Sie *MATX*.

[EXIT]

```
x: 0,2000 i0,0008
MATA MATB MATX
```

Definieren Sie *MATA* und *MATX* als reelle Matrizen; verlassen Sie anschließend die MATRIX Applikation.

[RCL] [MATA] ■[COMPLEX]
[STO] [MATA]
[RCL] [MATX] ■[COMPLEX]
[STO] [MATX]
[EXIT] [EXIT]

```
Y: [ 4x1 Matrix ]
x: [ 4x1 Matrix ]
```

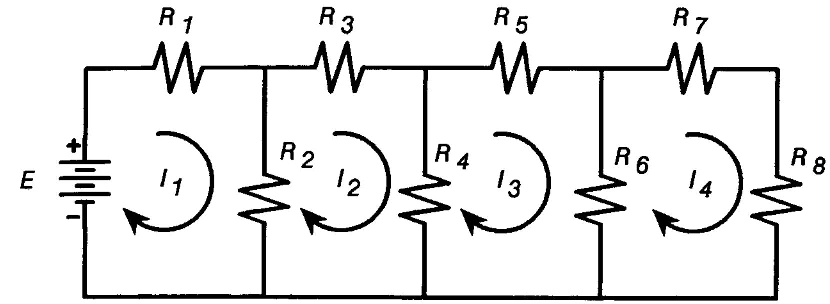
Verwenden des Lösers für lineare Gleichungssysteme

In den Beispielen des vorangehenden Abschnitts wurden die Maschenströme I_1 bis I_4 durch Division der Konstantenmatrix *MATB* durch die Koeffizientenmatrix *MATA* berechnet. Beim Bearbeiten dieser Beispiele waren Sie auf die spezifische Lösung der Maschenströme in der Ergebnismatrix *MATX* eingeschränkt.

Im folgenden Beispiel ist der Löser und eine Matrixdivision auszuführen, um den Wert eines Elements der *Koeffizientenmatrix*, *MATA*, zu berechnen; dabei sind bekannt:

- Die Werte aller anderen Elemente der Koeffizientenmatrix.
- Die Werte aller Elemente der Konstantenmatrix.
- Eine definierte Beziehung zwischen zwei Werten der Ergebnismatrix.

Beispiel: Berechnen eines Elements der Koeffizientenmatrix über den Löser. Betrachten Sie nochmals das ohmsche Netzwerk aus dem vorherigen Abschnitt.



Berechnen Sie die Größe des Widerstands R_1 so, daß der Maschenstrom I_1 um *20 A größer* als Maschenstrom I_2 ist ($I_1 = I_2 + 20$), wobei $E = 40$ V und R_2 bis $R_8 = 1\,\Omega$.

Die vorgegebenen Bedingungen führen zu folgender Matrix-Gleichung.

$$\begin{bmatrix} R & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} I_2 + 20 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} 40 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Teil 1. Schreiben Sie ein Programm für den Löser.

Programm: / **Kommentar:**

```
00 { 90-Byte Prgm }
01 LBL "SIMUL"
```

```
02 MVAR "R"
03 MVAR "ZEILE"
04 MVAR "SPALT"
05 MVAR "D"
```

Zeile 02–05: Definition der Variablen *R*, *ZEILE*, *SPALT* und *D*.

```
06 INDEX "MATA"
07 RCL "ZEILE"
08 RCL "SPALT"
09 STOIJ
10 RCL "R"
11 STOEL
```

Zeile 06–11: Indiziere die Koeffizientenmatrix und stelle den Indexzeiger auf das Element, das durch die momentanen Werte von *ZEILE* und *SPALT* (Zeile 05–08) spezifiziert ist. Speichere den momentanen Wert von *R* (von Ihnen anfangs als Anfangsnäherungen vorgegeben und dann iterativ durch den Löser ermittelt) im spezifizierten Element (Zeile 09–10).

```
12 RCL "MATB"
13 RCL÷ "MATA"
14 STO "MATX"
```

Zeile 12–14: Berechne *MATX*. *MATA* enthält den momentanen Wert von *R* im spezifizierten Element.

```
15 INDEX "MATX"
16 RCLEL
17 I+
18 RCLEL
19 RCL+ "D"
20 -
```

Zeile 15–20: Indiziere die soeben berechnete Ergebnismatrix (Zeile 14). Berechne $I_1 - (I_2 + D)$ (Zeile 15–20). Der Löser ermittelt iterativ die Werte für *R*, bis $I_1 - (I_2 + D) = 0$.

```
21 END
```


Teil 2. Rufen Sie die MATRIX Applikation auf und spezifizieren Sie ein Gleichungssystem mit 4 Unbekannten.

■[MATRIX] SIMQ 4 [ENTER]

```
x: 0,0000
MATA MATB MATX
```

Füllen Sie *MATA* mit den bekannten Koeffizienten auf. Element 1:1 enthält dabei den unbekannten Widerstandswert *R*; Sie können dessen momentanen Wert gespeichert lassen. (Die Tastenfolge für die ersten zwei Zeilen sind nachfolgend dargestellt.) Kehren Sie zum Hauptmenü zurück, nachdem alle Daten eingegeben wurden.

MATA →
1 [+/-] →
→ →
1 [+/-] →
3 →
1 [+/-] → ...

[EXIT]

```
x: 3,0000
MATA MATB MATX
```

Belegen Sie *MATB* mit den bekannten Konstanten und verlassen Sie danach die MATRIX Applikation.

MATB 40 ↓
0 ↓
0 ↓
0 ↓
[EXIT] [EXIT] [EXIT]

```
y: 0,0000
x: 40,0000
```

Rufen Sie die SOLVER Applikation auf und wählen Sie das Programm SIMUL.

■[SOLVER] SIMUL

```
x: 40,0000
R ZEILE SPALT D
```

Spezifizieren Sie Element 1:1 der Koeffizientenmatrix.

1 ZEILE 1 SPALT

```
SPALT=1,0000
R ZEILE SPALT D
```

Geben Sie den Wert 20 für *D* ein.

20 D

```
D=20,0000
R ZEILE SPALT D
```

Speichern Sie die Anfangsnäherungen 0 und 10 für *R* und berechnen Sie *R*.

0 R
10 R
R

```
R=1,6190
R ZEILE SPALT D
```

Überprüfen Sie, ob Element 1:1 der Koeffizientenmatrix (*R*) gleich 1,6190 ist.

■ [MATRIX] [▼]
EDITN MATA

```
1:1=1,6190
← OLD ↑ ↓ GOTO →
```

$R_1 = R - R_2 = 0{,}619\,\Omega$. Überprüfen Sie die Werte für I_1 und I_2.

[EXIT] EDITN MATX

```
1:1=32,3077
← OLD ↑ ↓ GOTO →
```

I_1 is 32,3077 A. Überprüfen Sie I_2

↓

```
2:1=12,3077
← OLD ↑ ↓ GOTO →
```

I_2 is 12,3077 A. Verlassen Sie die MATRIX Applikation.

[EXIT] [EXIT]

```
Y: 1,6190
X: 12,3077
```

Matrixoperationen in Programmen

Alle Matrixfunktionen – außer GOTO – sind programmierbar. In den folgenden Kapiteln über fortgeschrittene Statistikoperationen wird vermehrt von Matrixoperationen Gebrauch gemacht.

Das LIST Programm auf Seite 176 – 178 erlaubt Ihnen das Akkumulieren von Statistikdaten in einer Matrix über die gleiche Tastenfolge wie bei normaler Dateneingabe in die Summationsregister.

Das MLR Programm auf Seite 186 – 192 benutzt Matrix- und Statistikfunktionen zur Durchführung einer linearen Regression für Datensätze von drei unabhängigen Variablen. MLR erzeugt die Koeffizientenmatrix *MATA* und die Konstantenmatrix *MATB*. Es benutzt den Matrix-Editor zum Auffüllen der Matrizen mit Werten und führt danach eine Matrix-Division zur Berechnung der Ergebnismatrix *MATX* aus.

Das Programm PFIT auf Seite 218 – 222 bildet die Statistikdaten der momentan im X-Register gespeicherten Matrix ab; danach paßt es unter Verwendung des momentan spezifizierten Kurvenmodells eine Kurve an diese Daten an. Das Programm bildet die Kurve und die Datenpunkte ab, indem *x,y*-Datenpaare von komplexen Matrizen verwendet werden.

6

Statistik

In diesem Kapitel werden Ihnen fünf Programme für Statistikoperationen vorgestellt. Die Programme verwenden Statistikfunktionen, die in Kapitel 15 des Benutzerhandbuchs eingeführt wurden und beziehen die im vorherigen Kapitel vorgestellten Matrixoperationen mit ein.

- Drei Programme erlauben Ihnen das Akkumulieren von Daten in einer Matrix für später folgende Statistikoperationen:
  - LIST ermöglicht Ihnen das Auffüllen einer $n \times 2$ Matrix $\Sigma LIST$ mit x,y-Datenpaaren über die gleiche Tastenfolge wie beim Eingeben von Daten in die Summationsregister.
  - ΣFORM speichert eine $n \times m$ Matrix in $\Sigma LIST$ und redimensioniert $\Sigma LIST$ nach $nm \times 2$. Jedes Element der ursprünglichen Matrix wird zu einem Element in Spalte 2 von $\Sigma LIST$. Spalte 1 wird dabei mit Nullen aufgefüllt.
  - XWRT füllt Spalte 1 von $\Sigma LIST$ mit den x-Werten 1, 2, 3, ... , n auf, um eine Kurvenanpassung über ein lineares oder exponentielles Kurvenmodell zu ermöglichen.
- MLR berechnet eine multiple lineare Regression für zwei oder drei unabhängige Variablen, wobei die Σ+ Funktion und Matrixoperationen zur Anwendung kommen.
- PFIT bildet die x,y-Datenpaare von $\Sigma LIST$ ab und benutzt FCSTY zum Abbilden einer Kurve in Abhänigkeit des momentan gewählten Anpassungsmodells. (Eine kommentierte Liste von PFIT ist in Kapitel 7 auf Seite 218 – 222 enthalten.)

Statistische Berechnungen mit Listen

Um dem Rechner einen Satz von x,y-Datenpaaren für statistische Berechnungen vorzugeben, ist die Tastenfolge

y-Wert [ENTER] *x-Wert* [Σ+]

für jedes Datenpaar erforderlich. Die Summationskoeffizienten in den 6 (oder 13) Summationsregistern werden nach jedem Drücken von [Σ+] automatisch neu berechnet. Der Rechner behält jedoch *keine* Liste der individuellen Datenpaare bei.

Um die Summationsregister zu aktualisieren *und* eine Liste der x,y-Datenpaare zu erhalten, ist folgende Vorgehensweise erforderlich:

1. Erzeugen einer 2-spaltigen Matrix.
2. Verwenden des Matrix-Editors, um die Matrix mit den individuellen Datenpaaren aufzufüllen.
3. Kopieren der Matrix in das X-Register.
4. Ausführen von Σ+, um die Daten in Summationsregister zu akkumulieren.

(Sie haben dies bereits in Kapitel 5 im Abschnitt "Interaktives Verwenden von Indizierungs- und Statistikfunktionen" ausgeführt.)

Das LIST Programm. Das nachstehende Programm LIST ermöglicht Ihnen über die Tastenfolge

y-Wert [ENTER] *x-Wert* LIST+ (für jedes Datenpaar)

das Auffüllen einer 1- oder 2-spaltigen Matrix $\Sigma LIST$ mit x,y-Datenpaaren. LIST+ stellt dabei eines der drei Menüfelder dar, welche von LIST erzeugt werden. Beachten Sie, daß es sich hierbei um die gleiche Tastenfolge handelt, welche zum Eingeben von Statistikdaten in die Summationsregister benutzt wird.

Um LIST einzutippen:

1. Erzeugen Sie vor der Programmeingabe zuerst die Variable $\Sigma LIST$.
2. Weisen Sie vor der Programmeingabe die Funktionen J+ und J− dem CUSTOM Menü zu.
3. Erzeugen Sie die Labels LIST, LIST+, LIST− und CLIST, wenn Sie mit der Programmeingabe beginnen.

Hier eine kommentierte Liste des Programms LIST.

Programm:	**Kommentar:**
`00 { 197-Byte Prgm }` `01 LBL "LIST"`	

`02 CLMENU`
`03 "LIST+"`
`04 KEY 1 XEQ "LIST+"`
`05 "LIST-"`
`06 KEY 2 XEQ "LIST-"`
`07 "CLIST"`
`08 KEY 6 XEQ "CLIST"`
`09 MENU`
`10 STOP`
`11 GTO "LIST"`
	Zeile 02 − 11: Erzeuge und zeige Menütasten an.

`12 LBL "LIST+"`
`13 SF 25`
`14 XEQ I`
`15 FC?C 25`
`16 GTO 02`
`17 GROW`
`18 J-`
`19 J+`
`20 WRAP`
	Zeile 12 − 20: Falls $\Sigma LIST$ existiert, indiziere sie und erweitere sie um eine Zeile. Wenn nicht vorhanden, erzeuge sie und indiziere sie (über Zeile 32 − 42).

`21 LBL 00`
`22 STOEL`
`23 FS? 01`
`24 GTO 01`
`25 J+`
`26 X<>Y`
`27 STOEL`
`28 X<>Y`
	Zeile 21 − 28: Speichere x-Wert in Matrix. Ist Flag 01 gelöscht, speichere auch y-Wert.

`29 LBL 01`
`30 VIEW "ΣLIST"`
`31 RTN`
	Zeile 29 − 31: Ansehen der $\Sigma LIST$ Matrix.

`32 LBL 02`
`33 1`
`34 FS? 01`
`35 1`
`36 FC? 01`
`37 2`
`38 DIM "ΣLIST"`
`39 XEQ I`
`40 R↓`
`41 R↓`
`42 GTO 00`
	Zeile 32 − 42: Erzeuge eine 1- oder 2-spaltige Matrix $\Sigma LIST$.

```
43 LBL "LIST-"
44 SF 25
45 XEQ I
46 FC? 25
47 RTN
48 J-
49 RCLEL
50 FS? 01
51 GTO 03
52 J-
53 RCLEL
```

Zeile 43–53: Rufe die Elemente der letzten Zeile von *ΣLIST* in das X- (oder X- und Y-) Register zurück.

```
54 LBL 03
55 DELR
56 FS?C 25
57 GTO 01
```

Zeile 54–57: Lösche die letzte Zeile von *ΣLIST*.

```
58 LBL "CLIST"
59 CLV "ΣLIST"
60 RTN
```

Subroutine CLIST, Zeile 58–60: Lösche (*Clear*) die Variable *ΣLIST*.

```
61 LBL I
62 INDEX "ΣLIST"
63 RTN
```

Subroutine I, Zeile 61–63: Indiziere *ΣLIST*.

```
64 END
```


Anwendung von LIST:

1. Löschen Sie Flag 01 zur Bearbeitung von Statistikaufgaben mit zwei Variablen (x- und y-Werte). Setzen Sie Flag 01 zur Bearbeitung von Statistikaufgaben mit einer Variablen (nur x-Werte); das Programm macht *ΣLIST* zu einer 1-spaltigen Matrix.
2. Drücken Sie [XEQ] LIST.
3. Löschen Sie *ΣLIST* durch Drücken von CLIST.
4. Geben Sie die Datenpaare über die Tastenfolge *y-Wert* [ENTER] *x-Wert* LIST+ ein (für jedes Datenpaar).
5. Sie können das letzte Datenpaar löschen, indem Sie LIST- drücken.

Beispiel: Akkumulieren von Statistikdaten in einer Matrix. Verwenden Sie das Programm LIST zum Akkumulieren der folgenden x,y-Datenpaare in der Matrix *ΣLIST*. Berechnen Sie am Ende den Mittelwert der x- und y-Werte.

x-Wert	y-Wert
6	2
5	3
9	5
12	6
21	11
7	4

Löschen Sie Flag 01, um statistische Berechnungen mit zwei Variablen zu ermöglichen. Rufen Sie danach Programm LIST auf.

■[FLAGS] CF 01
[XEQ] LIST

```
x: 0,0000
LIST+ LIST-  CLIST
```

Löschen Sie *ΣLIST*.

CLIST

```
x: 0,0000
LIST+ LIST-  CLIST
```

Geben Sie das erste Datenpaar ein.

2 [ENTER] 6 LIST+

```
ΣLIST=[ 1x2 Matrix ]
LIST+ LIST-             CLIST
```

Tippen Sie das nächste Datenpaar ein.

3 [ENTER] 5 LIST+

```
ΣLIST=[ 2x2 Matrix ]
LIST+ LIST-             CLIST
```

Tippen Sie die restlichen Datenpaare ein (die Tastenfolge dazu ist hier nicht dargestellt). Verlassen Sie danach LIST.

[EXIT]

```
Y: 4,0000
X: 7,0000
```

Löschen Sie die Summationsregister. Rufen Sie *ΣLIST* in das X-Register zurück.

■[CLEAR] CLΣ
[RCL] ΣLIST

```
Y: 7,0000
X: [ 6x2 Matrix ]
```

Akkumulieren Sie die Daten von *ΣLIST* in die Summationsregister.

[Σ+]

```
Y: 7,0000
X: 6,0000
```

Berechnen Sie den Mittelwert der *x*- und *y*-Werte.

■[STAT] MEAN

```
X: 10,0000
Σ+ SUM MEAN WMN SDEV CFIT
```

Der Mittelwert aller *x*-Werte ist 10. Überprüfen Sie den Mittelwert der *y*-Werte.

[x≷y]

```
X: 5,1667
Σ+ SUM MEAN WMN SDEV CFIT
```

Verlassen Sie das STAT Menü.

[EXIT]

```
Y: 10,0000
X: 5,1667
```


Redimensionieren der *ΣLIST* Matrix in eine *nm* × 2 Matrix. Im vorherigen Beispiel wurde LIST zum Erzeugen einer 6 × 2 Matrix *ΣLIST* verwendet. Danach wurde *ΣLIST* in das X-Register zurückgerufen und Σ+ ausgeführt, um die *x,y*-Datenpaare aus der Matrix in die Summationsregister zu akkumulieren. *Um Σ+ auszuführen, während das X-Register eine Matrix enthält, muß die Spaltenanzahl dieser Matrix gleich 2 sein.* Wenn z.B. LIST zum Erzeugen einer *n* × 1 Matrix *ΣLIST* verwendet wurde (durch Setzen von Flag 01), so müssen Sie die Matrix vor der Ausführung von Σ+ redimensionieren.

Das nachstehende Programm ΣFORM redimensioniert *jede* Matrix *ΣLIST* der Dimension *n* × *m* in die Dimension *nm* × 2. Alle Elemente der Eingabematrix werden in die zweite Spalte gebracht. Die erste Spalte wird mit Nullen aufgefüllt.

```
00 { 58-Byte Prgm }
01 LBL "ΣFORM"

02 2
03 RCL "ΣLIST"
04 DIM?
05 ×
06 DIM "ΣLIST"
07 INDEX "ΣLIST"
08 1
09 ENTER
10 2
11 R<>R
12 RCL "ΣLIST"
13 TRANS
14 STO "ΣLIST"

15 END
```

Auffüllen von Spalte 2 von *ΣLIST* mit konstant zunehmenden ganzen Zahlen. Wenn Sie ein lineares oder exponentielles Kurvenmodell einem Datensatz von einvariabligen Statistikdaten anpassen möchten, können Sie nachstehendes Programm heranziehen. Das Programm XWRT belegt die erste Spalte der *ΣLIST* Matrix mit den Zahlen 1, 2, 3, ..., *n*. Wenn *ΣLIST* eine 1-spaltige Matrix darstellt, erzeugt XWRT automatisch eine neue Spalte.

Programm:	**Kommentar:**
`00 { 45-Byte Prgm }` `01 LBL "XWRT"`	

`02 RCL "ΣLIST"`
`03 DIM?`
`04 1`
`05 -`
`06 X≤0?`
`07 XEQ "ΣFORM"`
`08 INDEX "ΣLIST"`
	Zeile 02–08: Rufe *ΣLIST* zurück. Wenn die Matrix nur 1 Spalte hat, führe ΣFORM aus, um 2-spaltige Matrix zu erzeugen; indiziere sie anschließend.

`09 LBL 00`
`10 RCLIJ`
`11 X<>Y`
`12 ↓`
`13 FC? 76`
`14 GTO 00`
	Zeile 9–14: Fülle Spalte 1 mit den Werten 1, 2, 3, ..., n auf, bis das Ende der Spalte erreicht wird.

`15 END`	

Verwenden der Summationsfunktionen (Σ+, Σ- und CLΣ) in Programmen

Das in diesem Abschnitt vorgestellte Programm MLR benutzt die Σ+ Funktion und Matrixoperationen zum Durchführen einer mehrfachen linearen Regression für drei unabhängige Variablen.

MLR paßt einem Satz von Daten $\{ (x_i, y_i, z_i, t_i), i = 1, 2, ..., n \}$ eine lineare Gleichung der Form

$$t = a + bx + cy + dz$$

an. Dies erfolgt über die Methode der kleinsten Quadrate.

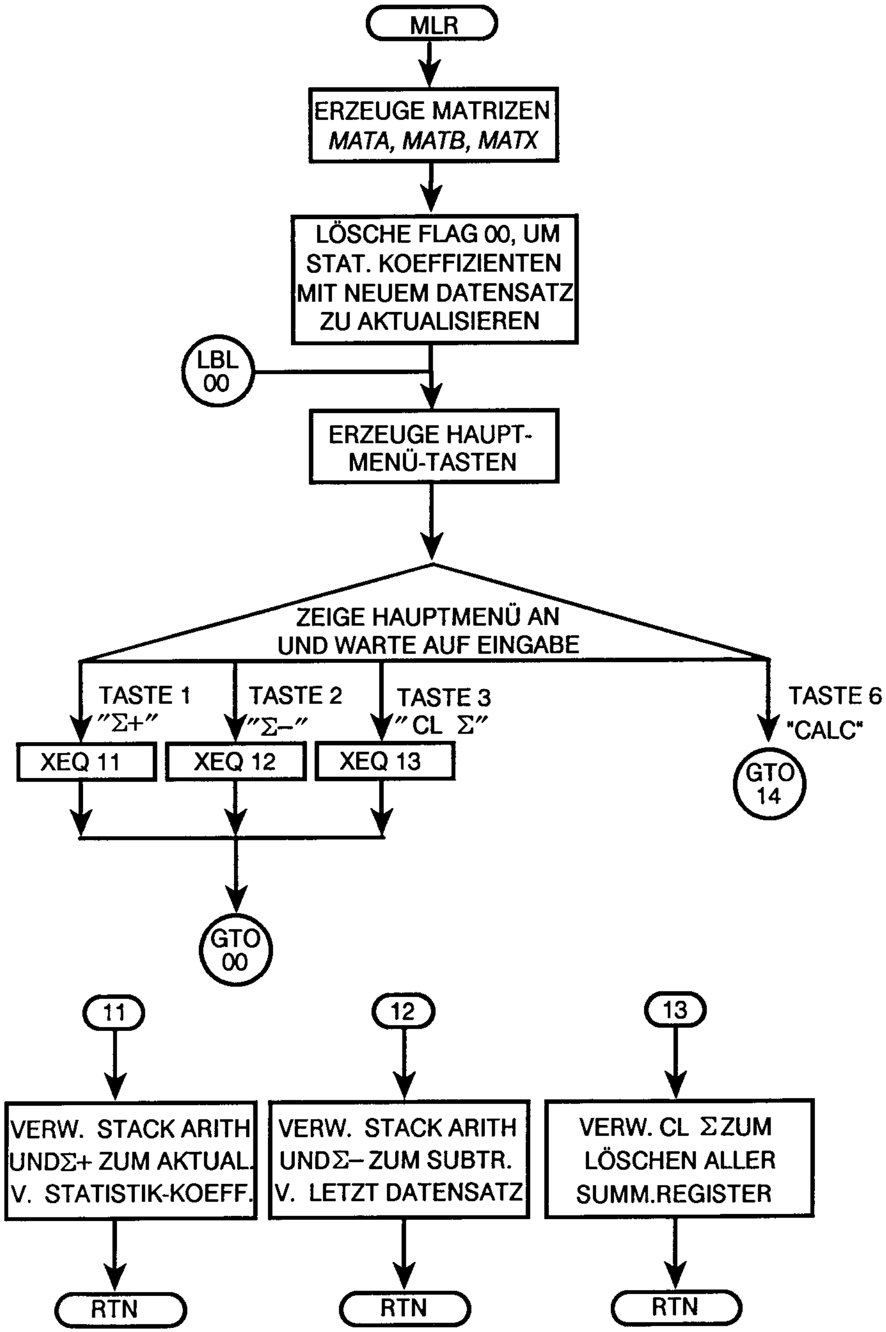
Die Regressionskoeffizienten a, b, c und d werden durch Lösen des nachstehenden Gleichungssystems berechnet:

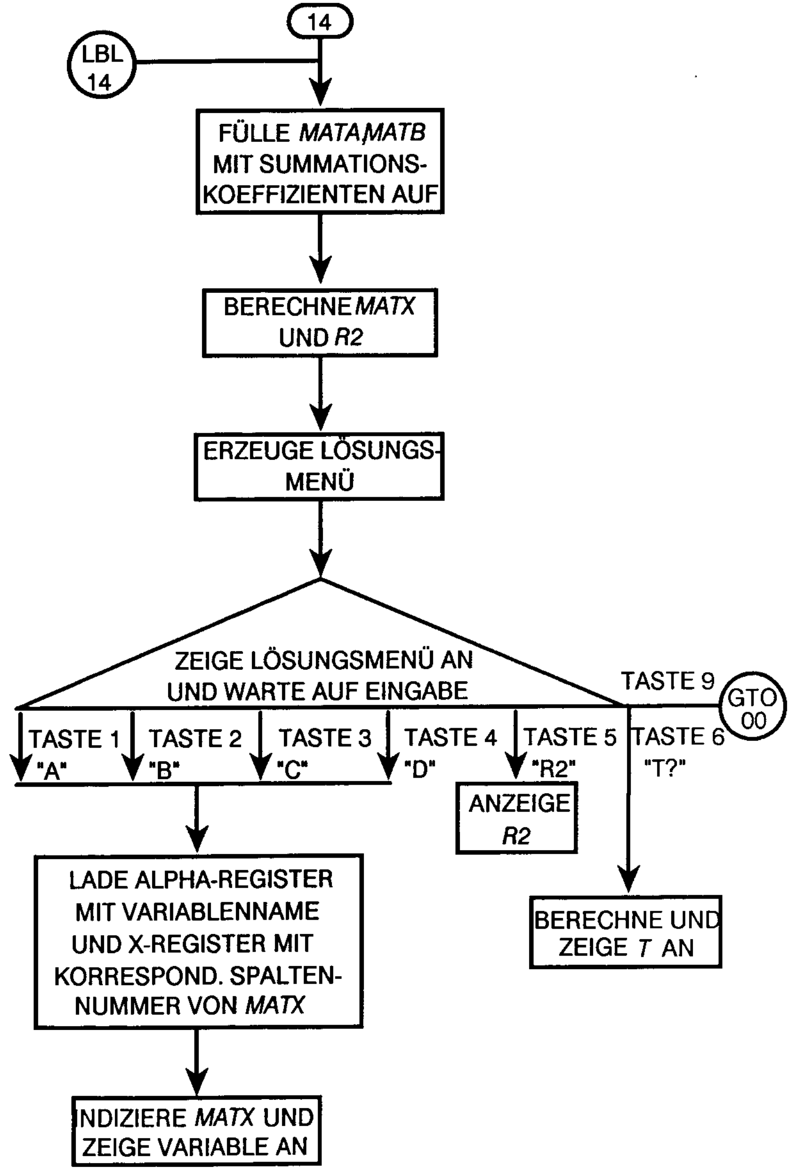
$$\begin{bmatrix} n & \Sigma x_i & \Sigma y_i & \Sigma z_i \\ \Sigma x_i & \Sigma (x_i)^2 & \Sigma x_i y_i & \Sigma x_i z_i \\ \Sigma y_i & \Sigma y_i x_i & \Sigma (y_i)^2 & \Sigma y_i z_i \\ \Sigma z_i & \Sigma z_i x_i & \Sigma z_i y_i & \Sigma (z_i)^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \Sigma t_i \\ \Sigma x_i t_i \\ \Sigma y_i t_i \\ \Sigma z_i t_i \end{bmatrix}$$

Das Bestimmtheitsmaß R^2 ist wie folgt definiert:

$$R^2 = \frac{a \Sigma t_i + b \Sigma x_i t_i + c \Sigma y_i t_i + d \Sigma z_i t_i - \frac{1}{n} (\Sigma t_i)^2}{\Sigma (t_i^2) - \frac{1}{n} (\Sigma t_i)^2}$$

Nachstehend ein Flußdiagramm für MLR.


MLR
ERZEUGE MATRIZEN MATA, MATB, MATX
LÖSCHE FLAG 00, UM STAT. KOEFFIZIENTEN MIT NEUEM DATENSATZ ZU AKTUALISIEREN
LBL 00
ERZEUGE HAUPT-MENÜ-TASTEN
ZEIGE HAUPTMENÜ AN UND WARTE AUF EINGABE
TASTE 1 "Σ+"
TASTE 2 "Σ–"
TASTE 3 "CL Σ"
TASTE 6 "CALC"
XEQ 11
XEQ 12
XEQ 13
GTO 14
GTO 00
11
12
13
VERW. STACK ARITH UNDΣ+ ZUM AKTUAL. V. STATISTIK-KOEFF.
VERW. STACK ARITH UNDΣ– ZUM SUBTR. V. LETZT DATENSATZ
VERW. CL ΣZUM LÖSCHEN ALLER SUMM.REGISTER
RTN
RTN
RTN


14
LBL 14
FÜLLE MATA,MATB MIT SUMMATIONS-KOEFFIZIENTEN AUF
BERECHNE MATX UND R2
ERZEUGE LÖSUNGS-MENÜ
ZEIGE LÖSUNGSMENÜ AN UND WARTE AUF EINGABE
TASTE 9
GTO 00
TASTE 1 "A"
TASTE 2 "B"
TASTE 3 "C"
TASTE 4 "D"
TASTE 5 "R2"
TASTE 6 "T?"
ANZEIGE R2
LADE ALPHA-REGISTER MIT VARIABLENNAME UND X-REGISTER MIT KORRESPOND. SPALTEN-NUMMER VON MATX
BERECHNE UND ZEIGE T AN
INDIZIERE MATX UND ZEIGE VARIABLE AN

Um MLR einzutippen:

1. Weisen Sie vor der Programmeingabe die Funktionen →, ↑, ←, ↓, I− und J+ dem CUSTOM Menü zu.
2. Erzeugen Sie vor der Programmeingabe die Variablen *MATA*, *MATB*, *MATX*, *R2* und *T*.

Hier eine kommentierte Liste des Programms.

Programm: / **Kommentar:**

```
00 { 462-Byte Prgm }
01 LBL "MLR"
```

```
02 REALRES
03 4
04 ENTER
05 1
06 DIM "MATX"
07 DIM "MATB"
08 4
09 ENTER
10 DIM "MATA"
11 CF 00
12 LINΣ
```

Zeile 02–12: Erlaube nur reelle Ergebnisse. Erzeuge die 4 × 1 Matrizen *MATX* und *MATB*. Erzeuge die 4 × 4 Matrix *MATA*. Lösche Flag 00 (Spezifizieren von Σ+ Modus). Spezifiziere Linear-Modus (Verwenden von sechs Summationskoeffizienten).

```
13 LBL 00
14 CF 21
15 CLMENU
16 "Σ+"
17 KEY 1 XEQ 11
18 "Σ-"
19 KEY 2 XEQ 12
20 "CLΣ"
21 KEY 3 XEQ 13
22 "CALC"
23 KEY 6 GTO 14
24 MENU
25 CLD
26 STOP
27 GTO 00
```

Zeile 13–27: Erzeuge und zeige die Menüfelder Σ+, Σ-, CLΣ und CALC an.


```
28 LBL 11
29 RCL× ST Z
30 FS? 00
31 +/-
32 STO+ 13
33 CLX
34 LASTX
35 RCL× ST T
36 FS? 00
37 +/-
38 STO+ 15
39 CLX
40 LASTX
41 R↓
42 RCL× ST Y
43 FS? 00
44 +/-
45 STO+ 14
46 CLX
47 LASTX
48 RCL× ST Z
49 FS? 00
50 +/-
51 STO+ 16
52 CLX
53 LASTX
54 R↓
55 ΣREG 07
56 FS? 00
57 RTN
58 Σ+
```

Subroutine 11, Zeile 28–58: *Emulation* von Σ+ (oder Σ−, falls Flag 00 gesetzt ist) zum Aktualisieren der folgenden Summationskoeffizienten: Σxz in R_{13}, Σxt in R_{15}, Σyz in R_{14}, Σyt in R_{16}. Führe Σ+ aus, um folgende Koeffizienten zu aktualisieren: Σz in R_{07}, Σz^2 in R_{08}, Σt in R_{09}, Σt^2 in R_{10}, Σzt in R_{11}, n in R_{12}.

```
59 LBL 01
60 CLX
61 LASTX
62 R↓
63 R↓
64 ΣREG 01
65 FS?C 00
66 RTN
67 Σ+
68 RTN
```

Subroutine 01, Zeile 59 – 68: Führe Σ+ aus, um folgende Koeffizienten zu aktualisieren: Σx in R_{01}, Σx^2 in R_{02}, Σy in R_{03}, Σy^2 in R_{04}, Σxy in R_{05} und n in R_{06}. (Beachten Sie, daß n auch in Subroutine 11 berechnet wird.)

```
69 LBL 12
70 SF 00
71 XEQ 11
72 Σ-
73 XEQ 01
74 Σ-
75 RTN
```

Subroutine 12, Zeile 69 – 75: Emulation von Σ− (setze Flag 00) zum Aktualisieren der in Subroutine 11 berechneten Koeffizienten. Führe Σ− aus, um die restlichen Koeffizienten zu aktualisieren.

```
76 LBL 13
77 ΣREG 11
78 CLΣ
79 ΣREG 07
80 CLΣ
81 ΣREG 01
82 CLΣ
83 RTN
```

Subroutine 13, Zeile 76 – 83: Führe CLΣ aus, um alle definierten Summationsregister zu löschen.

```
84 LBL 14
85 "Am Rechnen..."
86 AVIEW
87 0
88 STO× "MATA"
89 INDEX "MATA"
90 J+
91 RCL 01
92 →
93 RCL 03
```

Zeile 84 – 147, Berechnen der Koeffizienten a, b, c, d und R^2: Fülle *MATA* mit den x, y, z Summationskoeffizienten auf; fülle *MATB* mit Summationskoeffizient t auf. Berechne *MATX* (*MATB* ÷ *MATA*). Berechne R^2.


```
94 →
95 RCL 07
96 ↓
97 RCL 13
98 ←
99 RCL 05
100 ↓
101 J+
102 RCL 14
103 ↓
104 RCL "MATA"
105 TRANS
106 STO+ "MATA"
107 RCL 08
108 ↑
109 ←
110 RCL 04
111 ↑
112 ←
113 RCL 02
114 ↑
115 ←
116 RCL 06
117 STOEL
118 INDEX "MATB"
119 RCL 09
120 ↓
121 RCL 15
122 ↓
123 RCL 16
124 ↓
125 RCL 11
126 STOEL
127 RCL "MATB"
128 RCL÷ "MATA"
129 STO "MATX"
130 LASTX
131 TRANS
132 X<>Y
```

```
133 x
134 FNRM
135 RCL 09
136 X↑2
137 RCL÷ 06
138 -
139 LASTX
140 RCL 10
141 X<>Y
142 -
143 ÷
144 STO "R2"
145 CLD
146 FS? 55
147 SF 21
```

Zeile 148 – 164: Erzeuge das Ergebnismenü und zeige es an.

```
148 LBL 02
149 "A"
150 KEY 1 XEQ 21
151 "B"
152 KEY 2 XEQ 22
153 "C"
154 KEY 3 XEQ 23
155 "D"
156 KEY 4 XEQ 24
157 "R2"
158 KEY 5 XEQ 25
159 "T?"
160 KEY 6 XEQ 26
161 KEY 9 GTO 00
162 MENU
163 STOP
164 GTO 02
```

Subroutinen 21 – 25, Zeile 165 – 192: Zeige die berechneten Koeffizienten *a*, *b*, *c*, *d* und *R2* an. Falls PRON ausgeführt wurde, drucke die Koeffizienten (Zeile 187 und 191).

```
165 LBL 21
166 1
167 "a"
168 GTO 03
169 LBL 22
170 2
```



```
171 "b"
172 GTO 03
173 LBL 23
174 3
175 "c"
176 GTO 03
177 LBL 24
178 4
179 "d"
180 LBL 03
181 1
182 INDEX "MATX"
183 STOIJ
184 RCLEL
185 ⊢"="
186 ARCL ST X
187 AVIEW
188 RTN
189 LBL 25
190 RCL "R2"
191 VIEW "R2"
192 RTN
```

Subroutine 26, Zeile 193 – 205: Vorhersage von *T*, basierend auf den berechneten Koeffizienten *a*, *b*, *c* und *d*. Zeige *T* an und, falls PRON ausgeführt wurde, drucke *T*.

```
193 LBL 26
194 INDEX "MATX"
195 XEQ 04
196 XEQ 04
197 XEQ 04
198 +
199 +
200 I-
201 RCLEL
202 +
203 STO "T"
204 VIEW "T"
205 RTN
```

```
206 LBL 04
207 I-
208 RCLEL
209 RCL× ST T
210 RTN

211 END
```

Subroutine 04, Zeile 206–210: Berechne die Terme *bx*, *cy* und *dz*.

Anwendung von MLR:

1. Drücken Sie [XEQ] MLR
2. Drücken Sie CLΣ zum Löschen der Summationsregister.
3. Geben Sie jeden Datensatz unter Verwendung der Tastenfolge *t-Wert* [ENTER] *z-Wert* [ENTER] *y-Wert* [ENTER] *x-Wert* Σ+ ein.
4. Drücken Sie CALC .
5. Drücken Sie die korrespondierenden Menütasten, um die Werte der Variablen *a*, *b*, *c*, *d* und *R2* anzusehen.
6. Zur Vorhersage (-Berechnung) von *T* ist die Tastenfolge *z-Wert* [ENTER] *y-Wert* [ENTER] *x-Wert* T? zu benutzen.
7. Drücken Sie [EXIT], um zum Hauptmenü zurückzukehren.

Beispiel: Lineare Regression für drei unabhängige Variablen. Bestimmen Sie die Regressionsgleichung für folgende Daten:

i	**1**	**2**	**3**	**4**	**5**
x_i	7	1	11	11	7
y_i	25	29	56	31	52
z_i	6	15	8	8	6
t_i	60	52	20	47	33

Führen Sie MLR aus.

[XEQ] MLR

```
x: 4,0000
Σ+ Σ- CLΣ CALC
```


Löschen Sie die Summationsregister. Geben Sie den ersten Datensatz ein, beginnend mit dem *t*-Wert.

CLΣ
60 [ENTER] 6 [ENTER] 25 [ENTER]
7 Σ+

```
x: 1,0000
Σ+ Σ- CLΣ CALC
```

Geben Sie den zweiten Datensatz ein.

52 [ENTER] 15 [ENTER] 29 [ENTER]
1 Σ+

```
x: 2,0000
Σ+ Σ- CLΣ CALC
```

Geben Sie die restlichen Daten ein (die erforderliche Tastenfolge hierfür ist nicht dargestellt). Berechnen Sie anschließend die Regressionskoeffizienten und das Bestimmtheitsmaß.

CALC

```
x: 0,9989
A B C D R2 T?
```

Überprüfen Sie den Wert von *a*.

A

```
a=103,4473
A B C D R2 T?
```

Überprüfen Sie den Wert von *b*.

B

```
b=-1,2841
A B C D R2 T?
```

Überprüfen Sie den Wert von *c*.

C

```
c=-1,0369
A B C D R2 T?
```

Überprüfen Sie den Wert von *d*.

D

```
d=-1,3395
A B C D R2 T?
```

Überprüfen Sie den Wert von R^2.

R2

```
R2=0,9989
A B C D R2 T?
```

Berechnen Sie T (der vorhergesagte Wert von t bei gegebenen Werten für x, y und z). Verwenden Sie die Werte von Datensatz 4.

8 [ENTER] 31 [ENTER] 11 T?

```
T=46,4616
A B C D R2 T?
```

(Der tatsächliche Wert von t in Datensatz 4 ist 47.) Kehren Sie zum Hauptmenü zurück und löschen Sie den Inhalt der Statistikregister.

[EXIT] CLΣ

```
x: 46,4616
Σ+ Σ- CLΣ CALC
```

Verlassen Sie MLR.

[EXIT]

```
y: 11,0000
x: 46,4616
```

Kurvenanpassung über Programme

Die Kurvenanpassungsfunktionen FCSTX, FCSTY, SLOPE, YINT, CORR, LINF, LOGF, EXPF, PWRF und BEST lassen sich auch programmieren.

Beziehen Sie sich auf das Programm PFIT auf Seite 218 – 222 im folgenden Kapitel. PFIT benutzt FCSTY in Zeile 89, um einen y-Wert, basierend auf dem momentan gewählten Statistikmodell, für jeden der 110 x-Werte vorherzusagen. Im Anschluß daran wird eine Kurve der 110 Datenpaare abgebildet.

7

Grafische Darstellung

In diesem Kapitel sind folgende Themen behandelt:

- Erzeugen eines Grafik-Musters.
- Abbilden mehrerer Funktionen.
- Grafische Darstellung von Statistikdaten einer komplexen Matrix.

Grafiken

Das in diesem Abschnitt vorgestellte Programm HPLOGO benutzt die XTOA und AGRAPH Funktionen, um das Firmenzeichen von Hewlett-Packard in der Mitte der Anzeige abzubilden.

Um HPLOGO einzutippen:

1. Weisen Sie die Funktionen XTOA, CLA, ARCL und XEQ dem CUSTOM Menü zu.
2. Erzeugen Sie die Variable *BLOCK*.

Hier eine kommentierte Programmliste.

Programm: / **Kommentar:**

```
00 { 447-Byte Prgm }
01 LBL "HPLOGO"
```

```
02 CLLCD
03 CF 34
04 CF 35
```

Zeile 02–04: Lösche Anzeigeinhalt. Lösche Flag 34 und 35, um mit Hilfe von AGRAPH die in die Anzeige übertragenen Grafiken mit bereits angezeigten Grafiken zu mischen. (Die obere und untere Hälfte des Firmenzeichens werden getrennt erzeugt und später in der Anzeige zusammengefügt.)

```
05 XEQ "OBEN"
06 1
07 ENTER
08 40
09 AGRAPH
```

Zeile 05–09: Rufe Subroutine OBEN auf, um die obere Hälfte des Firmenzeichens zu erzeugen. Anzeige dieser Hälfte, beginnend mit Pixel (1, 40).

```
10 XEQ "UNTEN"
11 9
12 ENTER
13 40
14 AGRAPH
15 RTN
```

Zeile 10–15: Rufe Subroutine UNTEN auf, um die untere Hälfte des Firmenzeichens zu erzeugen. Anzeige dieser Hälfte, beginnend mit Pixel (9, 40).

```
16 LBL "OBEN"
17 CLA
18 255
19 XTOA
20 XTOA
21 XTOA
22 XTOA
23 XTOA
24 XTOA
25 ASTO "BLOCK"
26 CLA
27 254
```

Subroutine OBEN, Zeile 16–91: Erzeuge den Alpha-String, der die obere Hälfte des Firmenzeichens darstellt. (Beginne mit der Darstellung eines 8 × 6 Blocks von Ein-Pixel und speichere diesen String in der Variablen *BLOCK*.)


```
28 XTOA
29 ARCL "BLOCK"
30 255
31 XTOA
32 63
33 XTOA
34 15
35 XTOA
36 7
37 XTOA
38 XTOA
39 3
40 XTOA
41 1
42 XTOA
43 129
44 XTOA
45 224
46 XTOA
47 120
48 XTOA
49 62
50 XTOA
51 39
52 XTOA
53 161
54 XTOA
55 224
56 XTOA
57 96
58 XTOA
59 0
60 XTOA
61 1
62 XTOA
63 129
64 XTOA
65 225
66 XTOA
67 97
68 XTOA
```

```
69 33
70 XTOA
71 XTOA
72 35
73 XTOA
74 163
75 XTOA
76 231
77 XTOA
78 103
79 XTOA
80 15
81 XTOA
82 31
83 XTOA
84 63
85 XTOA
86 ARCL "BLOCK"
87 255
88 XTOA
89 254
90 XTOA
91 RTN
```

Subroutine UNTEN, Zeile 92–156: Erzeuge den Alpha-String, welcher die untere Hälfte des Firmenzeichens darstellt.

```
92 LBL "UNTEN"
93 CLA
94 127
95 XTOA
96 ARCL "BLOCK"
97 255
98 XTOA
99 252
100 XTOA
101 240
102 XTOA
103 224
104 XTOA
```



```
105 XTOA
106 192
107 XTOA
108 198
109 XTOA
110 135
111 XTOA
112 129
113 XTOA
114 0
115 XTOA
116 XTOA
117 6
118 XTOA
119 7
120 XTOA
121 129
122 XTOA
123 224
124 XTOA
125 120
126 XTOA
127 30
128 XTOA
129 7
130 XTOA
131 5
132 XTOA
133 132
134 XTOA
135 XTOA
136 XTOA
137 198
138 XTOA
139 199
140 XTOA
141 225
142 XTOA
143 224
```

```
144 XTOA
145 240
146 XTOA
147 248
148 XTOA
149 252
150 XTOA
151 ARCL "BLOCK"
152 255
153 XTOA
154 127
155 XTOA
156 RTN

157 END
```

Beispiel: Erzeugen eines Firmenzeichens. Bilden Sie das Firmenzeichen von Hewlett-Packard in der Anzeige ab. Falls Sie über einen Drucker verfügen, können Sie HPLOGO modifizieren, um den Ausdruck des Firmenzeichens zu ermöglichen; drucken Sie es anschließend.

Führen Sie HPLOGO aus.

[XEQ] HPLO

Fügen Sie die Anweisung PRLCD nach Zeile 14 von HPLOGO ein, um das Firmenzeichen ausdrucken zu können.

■[PRGM] ■[GTO] [·] 14 [ENTER]
■[PRINT] [▼] PRLCD [EXIT]

```
Y: 9,0000
x: 40,0000
```

Drucken Sie das Firmenzeichen.

■[PRINT] [▲] PON
[XEQ] HPLO

Verwenden von binären Daten zum Erzeugen eines Grafik-Musters. Das Firmenzeichen aus dem vorherigen Beispiel wurde erzeugt, indem der Spalten-Druckwert für jede der 91 Spalten berechnet werden mußte – ein zeitaufwendiges Verfahren. Das nachstehende Programm BINDATA berechnet den Spalten-Druckwert, wenn Sie die korrespondierende Folge der Binärzahlen eines Spaltenmusters eingeben.

Programm: / **Kommentar:**

```
00 { 60-Byte Prgm }
01 LBL "BINDATA"
```

```
02 CF 34
03 CF 35
04 BINM
```

Zeile 02–04: Lösche Flag 34 und 35. Spezifiziere Binär-Modus.

```
05 LBL 00
06 CLX
07 STOP
08 "     "
09 126
10 X<>Y
11 X>Y?
12 GTO 01
13 """
14 XTOA
15 ├""  "
```

Zeile 05–15: Lösche das X-Register und unterbreche das Programm zur Eingabe binärer Daten (Zeile 06–07). Erzeuge einen Alpha-String mit fünf Leerzeichen (Zeile 08). Abfrage, ob das binäre Datum (konvertiert in Dezimalwert) größer als 126 ist. Falls ja, springe zu Label 01, ansonsten schließe das korrespondierende Alphazeichen (aus HP-42S Zeichensatz) in Anführungszeichen ein und füge zwei Leerzeichen hinzu.

```
16 LBL 01
17 AIP
18 AVIEW
19 CLA
20 XTOA
21 1
22 ENTER
23 66
24 AGRAPH
25 GTO 00
```

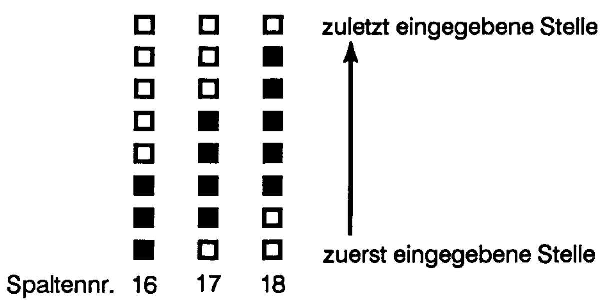
Zeile 16–25: Hänge die Zahl im X-Register (dezimales Äquivalent des Binärwerts) dem Alpha-Register an und zeige den momentanen Inhalt des Alpha-Registers an (Zeile 17–18). (Das Alpha-Register enthält den Dezimalwert des korrespondierenden Binärwerts. Ist die Zahl kleiner als 128, so enthält das Alpha-Register auch das korrespondierende HP-42S Zeichen, eingeschlossen in Anführungszeichen.) Erzeuge das zugehörige Spaltenmuster und zeige es an, beginnend mit Pixel (1, 66) (Zeile 20–24). Kehre zu Label 00 zurück, um nächste Eingabe zu verarbeiten.

```
26 END
```

Anwendung von BINDATA:

1. Ein Ein-Pixel hat den Wert 1; ein Aus-Pixel hat den Wert 0.
2. Geben Sie die Stellen ein; beginnen Sie dabei am unteren Ende der Spalte.
3. Wenn Sie z.B. nur sechs Stellen eingeben, werden die unteren zwei Stellen als Nullen interpretiert.
4. Drücken Sie zum Abschluß der Dateneingabe [R/S], um die Berechnung anzuzeigen. Geben Sie nach Anzeige der Ergebnisse einfach die nächste Zahlenfolge ein.

Beispiel: Verwenden von Binärwerten zum Erzeugen eines Firmenzeichens. Die Spalten 16–18 aus dem vorherigen Beispiel des Hewlett-Packard Firmenzeichens haben folgendes Pixel-Muster:



Verwenden Sie BINDATA zum Berechnen des Spalten-Druckwerts jeder Spalte.

Starten Sie das Programm.

[XEQ] BINDA

```
x: 0
A...F HEXM DECM OCTM BIN▪ LOGIC
```

Geben Sie den Binärwert für Spalte 16 ein.

11100000 [R/S]

```
224
```


Der Spalten-Druckwert für Spalte 16 ist 224. Es gibt kein zugehöriges Alphazeichen. Das Spaltenmuster befindet sich rechts in der Anzeige. Geben Sie nun den Binärwert für Spalte 17 ein.

01111000 [R/S]

```
"x"  120
```

Der Spalten-Druckwert für Spalte 17 ist 120. Das zugehörige HP-42S Zeichen ist "x". (Sie können daher entweder 120 in das X-Register übernehmen und XTOA ausführen, *oder* das Zeichen "x" im Alpha-Register akkumulieren.) Das Spaltenmuster befindet sich rechts in der Anzeige. Geben Sie nun den Binärwert für Spalte 18 ein.

00111110 [R/S]

```
">"  62
```

Der Spalten-Druckwert für Spalte 18 ist 62. Das zugehörige HP-42S Alphazeichen ist ">". Verlassen Sie nun das Programm.

[EXIT]

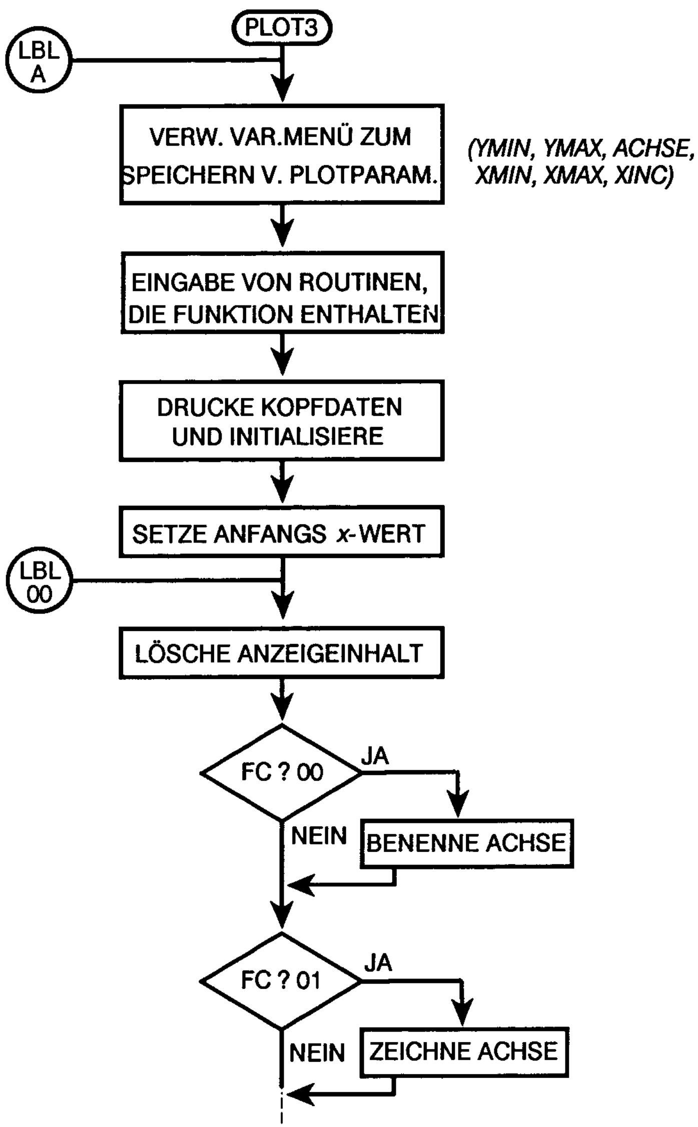
```
y: 1,0000
x: 0,0000
```

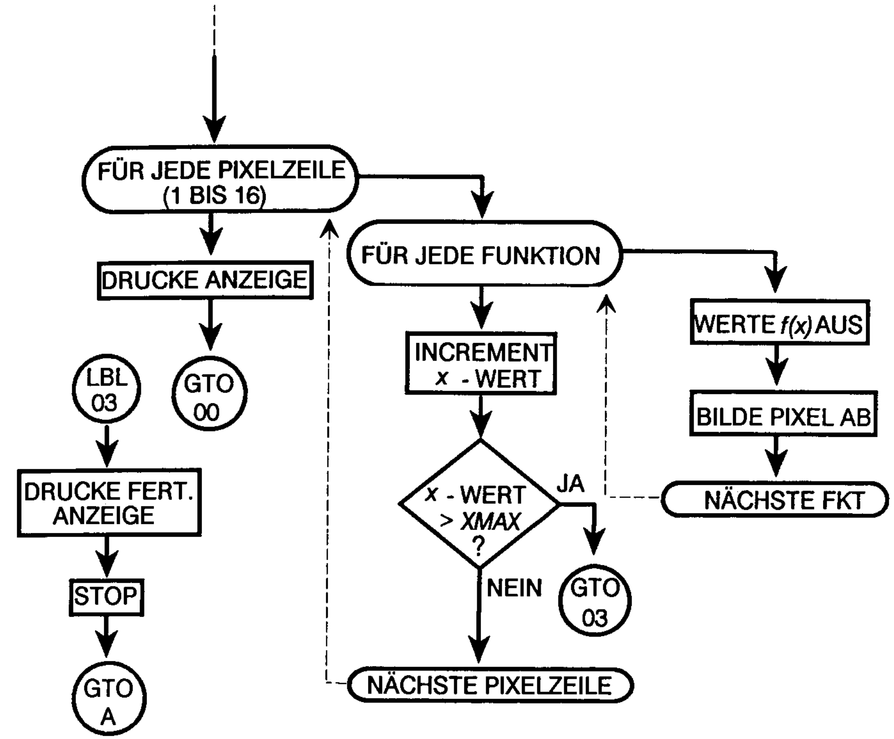
(Beziehen Sie sich auf die Zeichensatz-Tabelle in Ihrem Benutzerhandbuch (Anhang E) und beachten Sie, daß fünf der ersten 127 Zeichen nicht über das Tastenfeld des Rechners eingetippt werden können. Es handelt sich um die Zeichencodes 4, 6, 13, 27 und 30. Das Programm BINDATA zeigt Ihnen das Zeichen für jeden dieser Codes an. Da jedoch diese Zeichen nicht eintippbar sind, müssen Sie den korrespondierenden Zeichencode im X-Register akkumulieren und XTOA ausführen.)

Abbilden mehrerer Funktionen

Das in diesem Abschnitt vorgestellte Programm PLOT3 ermöglicht die gleichzeitige grafische Abbildung von bis zu drei Funktionen auf dem Infrarot-Taschendrucker HP 82240A. Es basiert auf dem Programm PLOT in Kapitel 10 des Benutzerhandbuchs. Wie in PLOT, so geben Sie hier die Routine vor, welche die abzubildende Funktion definiert. Allerdings können Sie in PLOT3 bis zu drei Routinen vorgeben.

Nachstehend ein Flußdiagramm für PLOT3.





Um PLOT3 einzutippen: Erzeugen Sie die Variablen *YMIN*, *YMAX*, *ACHSE*, *XMIN*, *XMAX*, *XINC*, *FKT1*, *FKT2* und *FKT3*, bevor Sie mit der Programmeingabe beginnen.

Hier nun eine kommentierte Programmliste von PLOT3:

Programm:	**Kommentar:**
`00 { 428-Byte Prgm }` `01 LBL "PLOT3"`	

`02 MVAR "YMIN"`
`03 MVAR "YMAX"`
`04 MVAR "ACHSE"`
`05 MVAR "XMIN"`
`06 MVAR "XMAX"`
`07 MVAR "XINC"`
	Zeile 02–07: Definition der Menüvariablen.

```
08 LBL A
09 VARMENU "PLOT3"
10 CF 34
11 CF 35
12 CLA
13 STOP
14 EXITALL
```

Zeile 08 – 14: Zeige das Menü an und unterbreche die Programmausführung zur Eingabe von Daten.

```
15 "FKT1"
16 XEQ 07
17 "FKT2"
18 XEQ 07
19 "FKT3"
20 XEQ 07
```

Zeile 15 – 20: Eingabeaufforderung für Funktionsnamen (Subroutine-Labels).

```
21 ADV
22 "Plot von:"
23 PRA
24 ADV
25 SF 12
26 RCL "FKT1"
27 XEQ 08
28 RCL "FKT2"
29 XEQ 08
30 RCL "FKT3"
31 XEQ 08
32 ADV
33 CF 12
34 PRV "YMIN"
35 PRV "YMAX"
36 PRV "ACHSE"
37 PRV "XMIN"
38 PRV "XMAX"
39 PRV "XINC"
40 ADV
41 "← YMIN"
42 ├"       YMAX →␊"
43 PRA
```

Zeile 21 – 43: Drucke Überschrift. (Der Alpha-String in Zeile 42 enthält 7 Leerzeichen vor *YMAX*.)


```
44 130
45 RCL "YMAX"
46 RCL- "YMIN"
47 ÷
48 STO 00
```

Zeile 44 – 48: Berechne den relativen *y*-Wert eines Pixels.

```
49 RCL "XMIN"
50 STO 01
```

Zeile 49 – 50: Speichere den ersten *x*-Wert.

```
51 LBL 00
52 CLLCD
53 FC? 00
54 XEQ 05
55 FC? 01
56 XEQ 06
57 1,016
58 STO 02
```

Zeile 51 – 58: Lösche die Anzeige. Ist Flag 00 gelöscht, benenne die *x*-Achse, ansonsten zeichne eine Achse. Erzeuge eine Schleifensteuerzahl für die 16 Anzeigezeilen.

```
59 LBL 01
60 49,051
61 STO 03
```

Zeile 59 – 61: Erzeuge eine Schleifensteuerzahl für die drei möglichen Funktionen. (Die Codes für die Zeichen "1", "2" und "3" sind 49, 50 bzw. 51. Routine 02 benutzt diese zum Erzeugen der Variablen.)

```
62 LBL 02
63 "FKT"
64 RCL 03
65 XTOA
66 ASTO ST X
67 RCL IND ST X
68 STR?
69 XEQ 04
70 ISG 03
71 GTO 02
72 RCL "XINC"
73 16
74 ÷
75 STO+ 01
```

Zeile 62 – 83: Erzeuge nacheinander die Alpha-Strings FKT1, FKT2 und FKT3. Rufe jeden String in das X-Register und rufe danach die *Variable* in das X-Register, welche mit dem String übereinstimmt. Abfrage, ob die Variable einen Alpha-String (einen Funktionsnamen) enthält (Zeile 62 – 68). Falls ja, zeige ein Pixel für jede Funktion an. Erhöhe den *x*-Wert. Nach Abschluß der Abbildung (wenn *x*-Wert > *XMAX*), springe zu Label 03. Wenn die momentane Anzeige abgeschlossen ist (Zeile 1 – 16 aufgefüllt), drucke den Anzeigeinhalt und beginne mit einer neuen Abbildung.

```
76 RCL "XMAX"
77 RCL 01
78 X>Y?
79 GTO 03
80 ISG 02
81 GTO 01
82 PRLCD
83 GTO 00
```

```
84 LBL 03
85 PRLCD
86 RTN
87 GTO A
```

Zeile 84 – 87: Drucke die Endanzeige und beende das Programm. (Zeile 87 erlaubt Ihnen durch Drücken von [R/S] das erneute Starten des Programms.)

```
88 LBL 04
89 RCL 01
90 XEQ IND ST Y
91 SF 24
92 RCL- "YMIN"
93 RCL× 00
94 1
95 +
96 CF 24
97 RCL 02
98 X<>Y
99 X>0?
100 PIXEL
101 RTN
```

Subroutine 04, Zeile 88 – 101: Werte die Funktion an der Stelle *x* aus und bilde den entsprechenden Pixel ab.

```
102 LBL 05
103 CF 21
104 CLA
105 ARCL 01
106 AVIEW
107 SF 21
108 RTN
```

Subroutine 05, Zeile 102 – 108: Benenne die *x*-Achse.


```
109 LBL 06
110 1
111 RCL "ACHSE"
112 RCL- "YMIN"
113 RCL× 00
114 +/-
115 1
116 -
117 PIXEL
118 +/-
119 2
120 -
121 "xxxxx"
122 AGRAPH
123 RTN
```

Subroutine 06, Zeile 109 – 123: Zeichne die Achse. (Der Alpha-String in Zeile 121 stellt fünf "Multiplikationszeichen" dar: Drücken Sie ■ [ALPHA] [×] [×] [×] [×] [×] [ENTER].)

```
124 LBL 07
125 CF 21
126 ASTO ST L
127 CLX
128 AVIEW
129 PSE
130 CLA
131 FS? 55
132 SF 21
133 SF 25
134 RCL IND ST L
135 CF 25
136 STR?
137 ARCL ST X
138 AON
139 CLD
140 STOP
141 AOFF
142 ALENG
143 X≠0?
144 ASTO ST X
145 STO IND ST L
146 RTN
```

Zeile 124 – 146: Eingabeaufforderung für Alpha-String (Funktionsname). Enthält die Variable bereits einen Alpha-String, so wird dieser String als Vorgabewert in das Alpha-Register zurückgerufen.

```
147 LBL 08
148 CLA
149 STR?
150 ARCL ST X
151 STR?
152 PRA
153 RTN

154 END
```

Subroutine 08, Zeile 147–153: Drucke die Funktionsnamen aus.

Anwendung von PLOT3:

1. Führen Sie PRON aus und schalten Sie den Infrarot-Taschendrucker ein.
2. Schreiben Sie eine Routine für jede abzubildende Funktion. Der momentane x-Wert befindet sich im X-Register, wenn das Programm die Routine der Funktion aufruft und muß nicht explizit in das X-Register zurückgerufen werden.
3. Spezifizieren Sie ALL als Anzeigeformat.
4. Starten Sie das Programm (drücken Sie [XEQ] PLOT3).
5. Geben Sie die Abbildungsparameter für das Programm vor. Spezifizieren Sie z.B. 20 für *YMIN*, indem Sie 20 YMIN drücken.
6. Nach Vorgabe der Abbildungsparameter ist [R/S] zu drücken.
   a. Bei Aufforderung ist der Name jeder Routine (der jeweiligen Funktion) in einer Variablen zu speichern. Um z.B. den Namen TAN für *FKT1* zu speichern, ist TAN [R/S] nach der ersten Eingabeaufforderung zu drücken.
   b. Haben Sie bereits eine Routine (deren Name) für eine Funktionsvariable vorgegeben, so wird dieser Name bei der Eingabeaufforderung angezeigt. Soll der Name beibehalten werden, so drücken Sie einfach [R/S].
   c. Möchten Sie nur zwei Funktionen abbilden, geben Sie nur die Namen für zwei Variablen vor. Lassen Sie für die dritte Variable das Alpha-Register leer (drücken Sie einfach [R/S]). Wird für die dritte Variable ein Name angezeigt, löschen Sie diesen mit Hilfe von [←] und drücken danach [R/S]. Möchten Sie nur eine Funktion abbilden, so geben Sie nur den Namen für eine Variable vor und lassen das Alpha-Register leer (oder löschen dieses) für die anderen zwei Variablen.

Beispiel: Abbilden mehrerer Funktionen. Verwenden Sie PLOT3 zum Abbilden der folgenden Funktionen:

1. $y = \sin x$
2. $y = 0{,}35(\ln x)(\cos x)$

Schreiben Sie zuerst die Routinen zur Beschreibung der Funktionen.

```
00 { 10-Byte Prgm }
01 LBL "SINUS"
02 SIN
03 END
```

Programm:

```
00 { 27-Byte Prgm }
01 LBL "LNCOS"
02 COS
03 LASTX
04 0,0001
05 +
06 LN
07 0,35
08 ×
09 ×
10 END
```

Kommentar:

Zeile 04–05: Stelle sicher, daß das Programm nicht versucht, ln (0) auszuwerten.

Stellen Sie ALL als Anzeigeformat ein. Führen Sie PRON aus. Löschen Sie die Flags 00 und 01, um die x-Achse zu benennen und zu zeichnen. Starten Sie PLOT3.

■[DISP] ALL
■[PRINT] [▲] PON
■[FLAGS] ■[FLAGS]
CF 00 CF 01 [EXIT]
[XEQ] PLOT3

```
x: 0
YMIN YMAX ACHSE XMIN XMAX XINC
```

Bilden Sie y-Werte zwischen −3 und 3 ab und wählen Sie die Achse bei $y = 0$.

3 [+/-] YMIN
3 YMAX
0 ACHSE

```
ACHSE=0
YMIN YMAX ACHSE XMIN XMAX XINC
```

Bilden Sie die *x*-Werte zwischen 0 und 720 in Schritten von je 60 pro Anzeige ab.

0 XMIN
720 XMAX
60 XINC

```
XINC=60
YMIN YMAX ACHSE XMIN XMAX XINC
```

Geben Sie den Programmnamen SINUS für die erste Funktionsvariable vor.

[R/S] SINUS

```
SINUS_
ABCDE FGHI JKLM NOPQ RSTUV WXYZ
```

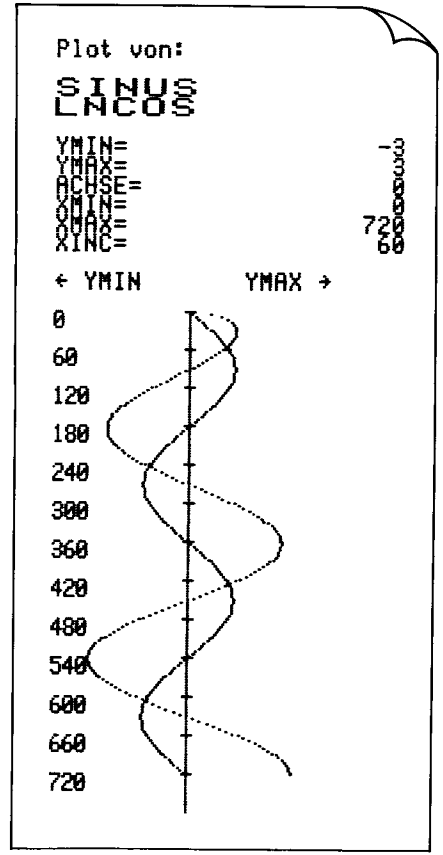
Geben Sie den Programmnamen LNCOS für die zweite Funktionsvariable vor.

[R/S] LNCOS

```
LNCOS_
ABCDE FGHI JKLM NOPQ RSTUV WXYZ
```


Lassen Sie das Alpha-Register für die dritte Funktion leer und starten Sie den Abbildungsvorgang. Nachstehend finden Sie den vom Drucker ausgegebenen Druckstreifen.

[R/S] [R/S]



Verlassen Sie das Programm. Stellen Sie wieder FIX 4 als Anzeigeformat ein.

[EXIT]
■[DISP] FIX 4 [ENTER]

```
Y: 720,0000
X: 723,7500
```

Abbilden von Daten einer komplexen Matrix

In vorangehenden Programmen kamen folgende Funktionen zur Anwendung:

- PIXEL, um individuelle Pixel in der Anzeige einzuschalten. Die Pixelnummer wurde dabei im X- und Y-Register (Zeilennummer in Y, Spaltennummer in X) angegeben.
- AGRAPH, um ein Grafik-Muster anzuzeigen. Die Spezifikation des Musters innerhalb der Anzeige erfolgte über die Vorgabe einer Pixelnummer im X- und Y-Register (Zeilennummer in Y, Spaltennummer in X).

PIXEL und AGRAPH *beziehen sich* auf die Zahlen im X- und Y-Register.

Die Effizienz dieser Funktionen wird noch dadurch erweitert, indem sie sich auf eine *komplexe Matrix* im X-Register beziehen können, wobei jedes Element der komplexen Matrix die Form

$$x\text{-Wert} + iy\text{-Wert}$$

besitzt. Befindet sich solch eine Matrix im X-Register, so schaltet PIXEL *jeden* Pixel in der Anzeige ein, wie durch die Matrixelemente vorgegeben. Betrachten Sie z.B. folgende komplexe Matrix:

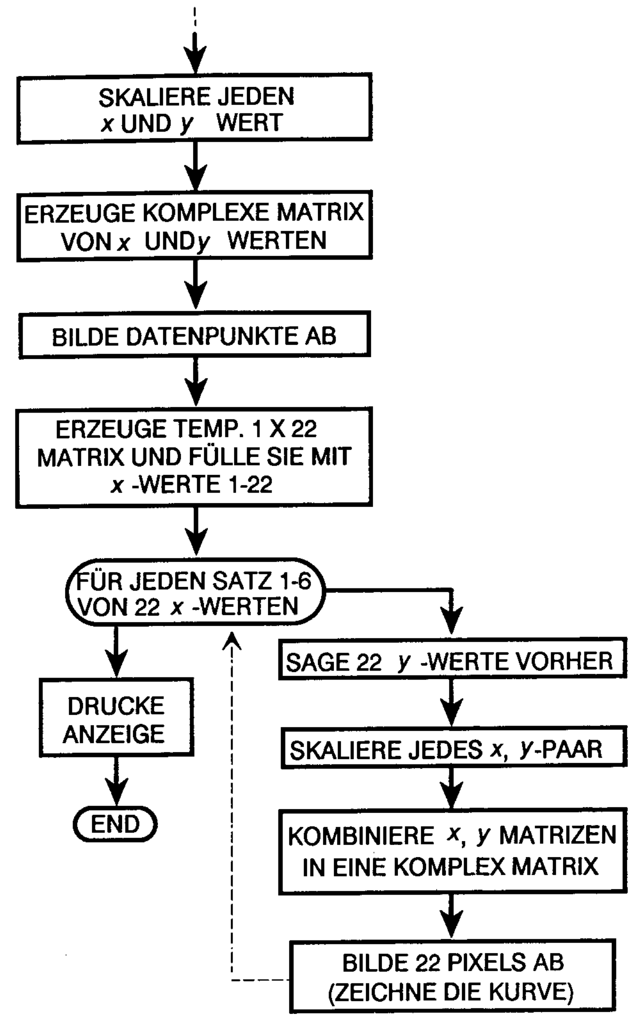
$$\begin{bmatrix} 1+i10 & 5+i20 \\ 10+i30 & 16+i40 \end{bmatrix}$$

Befindet sich diese Matrix im X-Register und Sie führen PIXEL aus, werden die Pixel (1, 10), (5, 20), (10, 30) und (16, 40) eingeschaltet.

Ähnlich dazu bildet AGRAPH das im Alpha-Register codierte Grafik-Muster an *jeder* Position in der Anzeige ab, entsprechend der Spezifikation durch die Matrixelemente.

Beachten Sie, daß PIXEL und AGRAPH sich auf die *Rechtecksform* der komplexen Matrix beziehen. Vor Eingabe von Zahlenwerten für die komplexe Matrix ist daher Rechtecksnotation als Winkelmodus zu spezifizieren.

Das in diesem Abschnitt enthaltene Programm PFIT bildet die individuellen Datenpaare von der reellen $n \times 2$ Matrix im X-Register ab; danach paßt es unter Verwendung des momentan gewählten Kurvenanpassungsmodells eine Kurve an diese Daten an und bildet diese ab. PFIT erzeugt eine komplexe Matrix und führt AGRAPH aus, um jeden Datenpunkt mit einem "+" Zeichen zu kennzeichnen. Danach erzeugt PFIT eine zweite komplexe Matrix und führt PIXEL aus, um die vorausberechnete Kurve abzubilden.



Um PFIT einzutippen:

1. Erzeugen Sie die Variable *DATAMTX*, bevor Sie mit der Programmeingabe beginnen.
2. Erzeugen Sie das Label MM, wenn Sie mit der Programmeingabe beginnen.

Nachstehend eine kommentierte Liste von PFIT.

Programm:	**Kommentar:**

```
00 { 295-Byte Prgm }
01 LBL "PFIT"
```

```
02 CF 34
03 CF 35
04 RECT
05 STO "DATAMTX"
06 INDEX "DATAMTX"
```

Zeile 02–06: Lösche Flag 34 und 35. Speichere Matrix aus X-Register in *DATAMTX* und indiziere *DATAMTX*.

```
07 XEQ "MM"
08 STO 02
09 -
10 128
11 ÷
12 STO 01
13 STO÷ 02
```

Zeile 07–13: Rufe Subroutine MM auf, um den kleinsten und größten *x*-Wert zu bestimmen. Berechnen Sie danach den Skalierungsfaktor für die *x*-Werte.

```
14 XEQ "MM"
15 X<>Y
16 STO 04
17 -
18 13
19 X<>Y
20 ÷
21 STO 03
22 STO× 04
23 2
24 STO- 04
25 STO- 02
```

Zeile 14–25: Rufe Subroutine MM auf, um den kleinsten und größten *y*-Wert zu bestimmen. Berechnen Sie danach den Skalierungsfaktor für die *y*-Werte.


```
26 CLLCD
27 RCL 04
28 X>0?
29 RCL- ST X
30 RCL 02
31 X>0?
32 CLX
33 PIXEL
```

Zeile 26–33: Bilde die Koordinatenachsen ab.

```
34 ΣREG 11
35 CLΣ
36 RCL "DATAMTX"
37 Σ+
```

Zeile 34–37: Speichere die Daten von *DATAMTX* in den Summationsregistern.

```
38 CLA
39 2
40 XTOA
41 7
42 XTOA
43 X<>Y
44 XTOA
```

Zeile 38–44: Erzeuge das "+" Zeichen (als Markierung für jeden Datenpunkt in der Abbildung benutzt).

```
45 RCL "DATAMTX"
46 TRANS
47 STO "DATAMTX"
48 INDEX "DATAMTX"
49 DIM?
50 1
51 X<>Y
52 GETM
53 DELR
54 RCL÷ 01
55 RCL- 02
```

Zeile 45–55: Dimensioniere die Matrix als $2 \times n$ Matrix und indiziere sie (Zeile 45–48). Erzeuge zwei $1 \times n$ Matrizen, wobei die Matrix im X-Register die *x*-Werte und die Matrix in *DATAMTX* die *y*-Werte enthält (Zeile 49–53). Konvertiere die *x*-Werte für Abbildungszwecke in Anzeigekoordinaten (Zeile 54–56).

```
56 RCL "DATAMTX"
57 RCL× 03
58 RCL- 04
59 COMPLEX
```

Zeile 56–59: Konvertiere die *y*-Werte in Anzeigekoordinaten. (Zeile 56–58). Konvertiere die Matrizen in X und Y in eine komplexe Matrix in X, wobei jedes Element *x*-Wert + *iy*-Wert (Zeile 57) entspricht.

```
60 1
61 ENTER
62 COMPLEX
63 -
64 AGRAPH
```

Zeile 60–64: Subtrahiere 1 + *i* von jedem Wert (um das Zentrum des "+" Zeichens für den Datenpunkt zu erhalten) (Zeile 60–63). Positioniere das Zentrum von "+" an den Koordinaten, die durch jedes Matrixelement definiert sind. Bilde die Datenpunkte ab (Zeile 64).

```
65 RCL "REGS"
66 1
67 ENTER
68 22
69 DIM "REGS"
```

Zeile 65–69: Rufe die Speicherregister-Matrix in das X-Register und redimensioniere sie zu einer temporären 1 × 22 Matrix.

```
70 21
71 LBL 01
72 STO IND ST X
73 DSE ST X
74 GTO 01
75 STO 00
```

Zeile 70–75: Fülle die temporäre Matrix mit den Werten 0 bis 21 auf.

```
76 R↑
77 RCL "REGS"
78 X<>Y
79 STO "REGS"
80 CLX
```

Zeile 76–80: Speichere die Daten aus den Registern zurück in *REGS* und lösche danach das X-Register.


```
81 6
82 X<>Y
83 1
84 +
```

Zeile 81–84: Erzeuge den Schleifenzähler 6,00000 im Y-Register. Ändere die Werte in der temporären Matrix auf 1 bis 22. (Diese Werte stellen den ersten Satz von 22 *x*-Werten dar.)

```
85 LBL 02
86 ENTER
87 RCL+ 02
88 RCL× 01
89 FCSTY
90 RCL× 03
91 RCL- 04
92 COMPLEX
93 PIXEL
94 COMPLEX
95 R↓
96 22
97 +
98 DSE ST Y
99 GTO 02
100 PRLCD
101 CLV "DATAMTX"
102 RTN
```

Zeile 85–102: Vorhersage der *y*-Werte für einen Satz von 22 *x*-Werten, erzeuge eine komplexe Matrix mit *x,y*-Datenpaaren und bilde jedes Datenpaar ab. Wiederhole dies für fünf weitere Sätze von *x*-Werten.

```
103 LBL "MM"
104 RCLEL
105 ENTER
106 LBL 09
107 I+
108 FS? 76
109 RTN
110 RCLEL
111 X<Y?
112 X<>Y
113 X<>Y
114 R↓
115 X>Y?
```

Subroutine MM, Zeile 103–120: Finde das größte und kleinste Element einer Matrixspalte, um eine Skalierung vornehmen zu können. Am Anfang der Subroutine wird die Matrix *DATAMTX* indiziert und der Indexzeiger wird an den Anfang einer Spalte gestellt. Am Ende der Subroutine befindet sich das kleinste Spaltenelement in X und das größte Element in Y; der Indexzeiger ist am Beginn (oben) der nächsten Spalte.

```
116 X<>Y
117 R↓
118 RCL ST Z
119 GTO 09
120 RTN

121 END
```

Anwendung von PFIT:

1. Wählen Sie ein Kurvenanpassungsmodell. Drücken Sie z.B. ■[STAT] CFIT MODL LINF.
2. Bringen Sie eine 2-spaltige reelle Matrix mit Datenpaaren in das X-Register.
3. Drücken Sie [XEQ] PFIT

Beispiel: Abbilden einer Kompressionskurve durch Anpassen eines Kurvenmodells and die Meßwerte. Viele Kompressionsvorgänge lassen sich mit Hilfe der Potenzkurve

$$P = aV^{-b}$$

in Korrelation bringen, wobei:

P = Druck
V = Volumen
$-b$ = polytropische Konstante

Geben Sie die folgenden Meßwerte für Druck und Volumen in die Matrix *ΣLIST* ein. Verwenden Sie danach PFIT zum Abbilden der Daten selbst sowie einer Potenzkurve an die Daten.

V	P
10	210
30	40
50	12
70	9
90	6,8

Führen Sie das Programm LIST aus. (Falls das Programm zwischenzeitlich gelöscht wurde, müssen Sie es erneut eintippen. Die Programmliste finden Sie im Abschnitt "Statistische Berechnungen mit Listen" in Kapitel 6.)

[XEQ] LIST

```
x: 0,0000
LIST+ LIST-  CLIST
```

Löschen Sie den Inhalt von *ΣLIST* und geben Sie danach die Daten ein.

CLIST
210 [ENTER] 10 LIST+
40 [ENTER] 30 LIST+
12 [ENTER] 50 LIST+
9 [ENTER] 70 LIST+
6,8 [ENTER] 90 LIST+

```
ΣLIST=[ 5x2 Matrix ]
LIST+ LIST-  CLIST
```

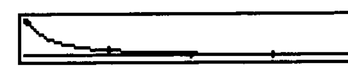
Verlassen Sie LIST. Rufen Sie *ΣLIST* in das X-Register zurück.

[EXIT]
[RCL] ΣLIST

```
y: 90,0000
x: [ 5x2 Matrix ]
```

Spezifizieren Sie eine Potenzkurve als Kurvenmodell und führen Sie PRON aus, falls Sie einen Drucker besitzen. Starten Sie anschließend das Programm PFIT.

■[STAT] CFIT MODL PWRF
(■[PRINT] [▲] PON)
■[TOP.FCN] XEQ PFIT



Verlassen Sie PFIT. Überprüfen Sie den Korrelationskoeffizienten.

[EXIT] CORR

```
x: -0,9939
FCSTX FCSTY SLOPE YINT CORR MODL
```

Der Korrelationskoeffizient ist $-0{,}9939$. Überprüfen Sie $-b$.

SLOPE

```
x: -1,6152
FCSTX FCSTY SLOPE YINT CORR MODL
```

$-b$ beträgt $-1{,}6252$. Verlasssen Sie das STAT Menü.

[EXIT] [EXIT]

```
y: -0,9939
x: -1,6152
```

Index

G

H

I

K

N

O

P

Q

R

X

Z