Benutzerhandbuch

SCEPTRE
Ein Programm zur Analyse und Simulation nichtlinearer elektrischer Netzwerke und dynamischer Systeme

Wolf-Rainer Novender*

Oktober 2018

©1989 – 2018. Alle Rechte vorbehalten.

*D-64625 Bensheim

Gesetzt mit LuaTeX, Version 1.07.0 (TEX Live 2018) und KOMA-Script

Inhaltsverzeichnis

1 Vorwort **1**
1.1 Leistungsüberblick . . . 1
1.2 Hard- und Softwarevoraussetzungen für Linux/UNIX . . . 2
1.3 Entwicklungsgeschichte . . . 2
1.4 Beispiel . . . 2
1.5 Hinweise zu diesem Handbuch . . . 4

I Programmbeschreibung (Volume I) **5**

2 SCEPTRE-Anweisungen **7**
2.1 Sprachvereinbarungen . . . 7
2.2 Vorbereitungen . . . 9
2.3 Eingabeformat . . . 11
2.4 Eingabestruktur . . . 11
2.5 ELEMENTS . . . 11
2.5.1 Allgemeine Zweipolform . . . 12
2.5.2 Gegeninduktivitäten . . . 14
2.5.3 Widerstandsabhängige Quellen . . . 14
2.5.4 Gesteuerte Stromquellen . . . 14
2.5.5 Zeitliche Ableitungen von Quellen . . . 15
2.5.6 Modellaufruf . . . 15
2.5.7 Elemente mit Toleranzgrenzen . . . 15
2.5.8 Quellen bei der Wechselstromanalyse . . . 15
2.6 DEFINED PARAMETERS . . . 16
2.7 OUTPUTS . . . 17
2.8 INITIAL CONDITIONS . . . 19
2.9 FUNCTIONS . . . 19
2.9.1 EQUATION-Anweisung . . . 19
2.9.2 TABLE-Anweisung . . . 20
2.10 RUN CONTROLS . . . 21
2.10.1 DC-Analyse . . . 21
2.10.2 AC-Analyse . . . 23
2.10.3 TR-Analyse . . . 23
2.10.4 DC- und TR-Analyse . . . 24
2.10.5 DC- und AC-Analyse . . . 25
2.10.6 AC- und TR-Analyse . . . 25
2.10.7 Ausgabesteuerung . . . 25
2.10.8 Wiederholungsläufe . . . 25
2.10.9 Ablaufsteuerung . . . 26
2.10.10 Testhilfen . . . 27
2.11 SENSITIVITY, MONTE CARLO, WORST CASE, OPTIMIZATION . . . 27

2.12 Modelle . . . 28
2.12.1 Modellbeschreibung . . . 29
2.12.2 Modellaufruf . . . 30
2.13 RERUN DESCRIPTION . . . 31
2.14 CONTINUE . . . 36
2.15 RE-OUTPUT . . . 36
2.16 END . . . 37

3 Allgemeine Hinweise 39
3.1 Fehlerdiagnose . . . 39
3.2 Einige Grundregeln . . . 39
3.3 Zeitabhängige Induktivitäten und Kapazitäten . . . 40
3.4 Tabellen . . . 40
3.5 Arithmetische Anweisungen . . . 40
3.6 Topologische Einschränkungen . . . 41
3.6.1 DC-, AC- und TR-Analyse . . . 41
3.6.2 DC-Analyse . . . 41
3.7 „Computational Delay“ . . . 41
3.8 Wahl des Integrationsverfahrens . . . 42
3.9 Definition abhängiger Quellen . . . 43
3.10 Strom- und Spannungsmessungen . . . 44
3.11 Verwendung eigener FORTRAN-Unterprogramme . . . 44
3.12 Die Vektornotation . . . 45
3.13 Wahl der Zustandsvariablen . . . 46
3.14 Berechnung von Integralen . . . 46
3.15 Simulationsstatistik . . . 46
3.16 Druckausgabe . . . 48
3.17 Anfangsbedingungen . . . 48
3.17.1 Arbeitspunkt . . . 48
3.17.2 Implizite Integration . . . 49
3.17.3 Wiederholungsläufe . . . 50
3.18 Periodische Funktionen . . . 50
3.18.1 Ideale Funktionen . . . 51
3.18.2 FPULSE . . . 52
3.18.3 FSINSQ . . . 53
3.19 Reservierte Variablennamen . . . 53
3.20 Wechselstromquellen . . . 54

4 Beispiele 55
4.1 A01: Inverter-Schaltung mit RC-Last . . . 55
4.2 A02: Emitterfolger mit Übertrager . . . 57
4.3 A03: Darlington-Schaltung (DC-Analyse) . . . 59
4.4 A04: Kleinsignal-Ersatzschaltung . . . 62
4.5 A05: Differentialgleichungssystem . . . 63
4.6 Toleranzanalysen einer Darlington-Schaltung . . . 64
4.6.1 A07: Monte-Carlo-Analyse . . . 65
4.6.2 A08: Empfindlichkeitsanalyse . . . 67
4.6.3 A09: ‘Worst Case’-Analyse . . . 70
4.6.4 A10: Optimierung . . . 72
4.7 A11: Kleinsignal-Ersatzschaltung (AC) . . . 73

4.8 A13: Elektro-hydraulischer Stellantrieb . . . 76

II Mathematische Beschreibung (Volume II) 81

5 Die Theorie hinter SCEPTRE 83
5.1 Überblick . . . 83
5.2 Genereller Lösungsansatz . . . 83
5.3 Transiente Lösung . . . 86
5.3.1 Matrix Operationen . . . 86
5.3.2 Lösungsschema . . . 88
5.3.3 Zeitliche Ableitungen von Quellen . . . 90
5.3.4 Linear abhängige Quellen . . . 90
5.3.5 Integrationsverfahren . . . 92
5.4 DC Analyse . . . 101
5.4.1 Die Newton-Raphson Methode . . . 101
5.4.2 Monte-Carlo . . . 106
5.4.3 Empfindlichkeit . . . 107
5.4.4 *Worst-Case* . . . 108
5.4.5 Optimierung . . . 110
5.5 AC Analyse . . . 112
5.5.1 Fall 1: nur unabhängige Quellen . . . 112
5.5.2 Fall 2: Schaltungen mit linearabhängigen Quellen . . . 115
5.5.3 Fall 3: Schaltungen, die die zeitlichen Ableitungen unabhängiger Quellen benötigen . . . 117
5.5.4 Fall 4: Schaltungen mit linear abhängigen Quellen, die zeitliche Ableitungen unabhängiger Quellen benötigen . . . 118

6 Progammablauf 119

III Benutzungsanleitung 127

7 Das Programmsystem SCEPTRE (Linux/UNIX-Version) 129
7.1 Programmorganisation . . . 129
7.2 Aufruf unter Unix . . . 130
7.3 Grafikschnittstellen . . . 131
7.3.1 Das Unterprogramm `INTPLT` . . . 131
7.3.2 Beispiel zur Ausgabe mittels ANSI Treibers . . . 132
7.3.3 Das Unterprogramm `POST` . . . 133
7.4 Auswertung der Ergebnisse (“Postprocessing”) . . . 135
7.5 Das Grafikprogramm NGP . . . 136
7.5.1 Übersicht der Kommandos . . . 136
7.5.2 NGP und GNUPLOT . . . 137
7.5.3 Konfigurationsdatei `ngp.rc` . . . 138
7.6 Fortsetzungsläufe (`CONTINUE`) . . . 139

IV Anhang 141

A Dioden- und Transistor-Ersatzschaltbilder 143
A.1 Diode . . . 143

A.2 pnp Transistor 144
A.3 npn Transistor 145

B Anhänge zu Vol. I [2] 147
B.1 Übersicht der SCEPTRE-Anweisungen 147
B.2 COMMON-Bereich `/CNTRLS/` 149
B.3 Programmkapazität 149

C Anhänge zu Vol. II [3] 151
C.1 Schwach besetzte Matrizen (*Sparse-Matrix*-Technik) 151
C.2 Assemblerroutinen für die Ein-/Ausgabe 152
C.3 Das *implicite* Integrationsverfahren 152
C.4 Herleitung der $\boldsymbol{B}$-Matrix 152
C.5 Auswertung der Diodenkennlinie bei der Berechnung der Anfangsbedingungen 153
C.6 Die Newton-Raphson-Methode 153
C.7 Identität der Gleichungen (5.79) und (5.80) 153
C.8 Adjungiertes Netzwerk 153

D Das Shell-Script `sceptre` 155

Abbildungsverzeichnis

1.1 Einfache Inverter-Schaltung . . . 3
1.2 SCEPTRE Darstellung der Inverter-Schaltung . . . 3

2.1 Spannungsverlauf . . . 20

3.1 Zeitmultiplex-Signal . . . 52
3.2 Die periodische Funktion FPULSE . . . 52
3.3 Die periodische Funktion FSINSQ . . . 53

4.1 Inverterschaltung und für SCEPTRE aufbereitetes Ersatzschaltbild . . . 55
4.2 Spannung `VCE` in Abhängigkeit von `JX` entsprechend `TABLE 1` . . . 56
4.3 Spannung `VC1` in Abhängigkeit von `JX` entsprechend `TABLE 1` . . . 57
4.4 Emitterfolger mitÜbertrager, Transistor-Ersatzschaltbild . . . 58
4.5 Spannungen `VL1` und `VL2` am Übertrager . . . 59
4.6 Darlington Schaltung . . . 60
4.7 Kleinsignal-Schaltung, Ersatzschaltbild mit h-Parametern . . . 62
4.8 Spannungsverlauf am Widerstand `RL1` . . . 63
4.9 Spannungsverlauf am Widerstand `RL2` . . . 64
4.10 Verlauf von `PX, PY, PZ` . . . 65
4.11 Kleinsignalschaltung, Transistor-Ersatzschaltbild . . . 74
4.12 Nyquist-Plot von `VCCT1` . . . 76
4.13 Schaltung des Regelkreises . . . 77
4.14 Magnetstrom `ILTORK` als Funktion der Zeit . . . 79
4.15 Regelgröße `PDIS` als Funktion der Zeit . . . 80

5.1 Zusammengesetzte B-Matrix . . . 84
5.2 Beispiel einer B-Matrix mit $B_{14}, B_{36} = 0, B_{25} \neq 0$ . . . 89
5.3 Kleinsignal-Transistorersatzschaltbild mit h-Parametern . . . 90
5.4 B-Matrix mit linear abhängigen Quellen . . . 91
5.5 Bedingung $0 < ah < 6$ für die numerische Stabilität des TRAP-Verfahrens . . . 95
5.6 Schaltung mit Kapazitätsschnittmenge . . . 104
5.7 Schaltung mit einer Induktivitätsmasche . . . 105
5.8 Knotenpunkt mit Kapazitäten und Stromquellen . . . 106
5.9 Zweidimensionales Beispiel zur *Worst-Case*-Analyse . . . 109
5.10 Änderung der Zielfunktion bei Änderung der unabhängigen Variablen . . . 112

6.1 System Flow Diagram . . . 120
6.2 Program Generator (EXEC1) Flow Diagram . . . 121
6.3 Circuit Description Processor (CDPRS) Flow Diagram . . . 122
6.4 Model Editor (MDEDIT) Flow Diagram . . . 123
6.5 Circuit Equation Generator (GENER8) Flow Diagram . . . 124
6.6 Rerun Processor and Data Generator (RRPRO) Flow Diagram . . . 125
6.7 Circuit Solution Executive (EXEC2) Flow Diagram . . . 126

7.1 Programm- und Datenfluss 129
7.2 Protokoll zur DA-Datei 135

A.1 SCEPTRE Diodenersatzschaltbild 143
A.2 SCEPTRE npn Transistor-Ersatzschaltbild 144
A.3 SCEPTRE pnp Transistor-Ersatzschaltbild 145

Tabellenverzeichnis

2.1 Klassifizierung der Elemente . . . 10
2.2 Länge der frei wählbaren Namen . . . 10
2.3 Beispiele für konsistente Maßeinheitensysteme . . . 10
2.4 Klassifizierung weiterer SCEPTRE-Größen . . . 11
2.5 Struktur der SCEPTRE Eingabedaten . . . 12
2.6 Festlegung von Strom und Spannung bei allgemeinen Zweipolen . . . 13
2.7 Abhängige Variablen . . . 28
2.8 Unabhängige Variablen . . . 28

5.1 Einteilung der Netzwerkelemente in Klassen . . . 85
5.2 B-Matrix für die TR-Analyse . . . 86
5.3 Das Verhältnis r/p für verschiedene Integrationsverfahren . . . 96
5.4 B-Matrix eines L-Baumes . . . 102
5.5 Empfindlichkeitsberechnung . . . 107

B.1 SEPTRE-Anweisungen und die dazugehörigen Variablen im COMMON-Block `CNTRLS` 147
B.2 Programmkapazität . . . 150

C.1 Vergleich der Rechenzeiten für Phase I zwischen FORTRAN-I/O and Assembler-I/O . 152

1 Vorwort

Das Programm SCEPTRE wurde zur Analyse elektrischer Netzwerke entwickelt unter besonderer Berücksichtigung nichtlinearer Effekte. Der Name steht für *System for Circuit Evaluation and Prediction of Transient Radiation Effects* [1, 2, 3].

1.1 Leistungsüberblick

Das Programm zeichnet sich durch folgende Eigenschaften aus:

- Gleichstromanalyse (DC) einschließlich Worst-Case-, Empfindlichkeits-, Monte-Carlo-Analyse und Optimierung,
- Transientanalyse (TR),
- Wechselstromanalyse (AC),
- Kombination der Analysearten (DC+TR, DC+AC),
- Methode der Zustandsvariablen;
- Automatisches Analyseprogramm: aus der Topologie und den Elementen des Netzwerkes erzeugt das Programm automatisch die Netzwerkgleichungen;
- Berechnung nichtlinearer Netzwerke: jedes nichtlineare Element kann durch ein beliebig kompliziertes Modell beschrieben werden. Netzwerkelemente können als Konstanten, in Form von Tabellen, algebraischen Ausdrücken bzw. in Abhängigkeit anderer Netzwerkgrößen vorliegen;
- Eingabe der Netzwerkdaten und Steueranweisungen im freien Format: Steueranweisungen in leicht merkbarer Form, keine Programmierkenntnisse erforderlich;
- Modellbibliothek: häufig benutzte Schaltungen oder Teilschaltungen (Ersatzschaltbilder) können permanent gespeichert und mit einer Anweisung in das zu untersuchende Netzwerk eingefügt werden;
- Arbeitspunktbestimmung: die Gleichstromanalyse kann in Verbindung mit der AC- oder TR-Analyse zur Bestimmung des Arbeitspunktes einer Schaltung kombiniert werden;
- Wiederholungsläufe: Analysen können mit veränderten Parametern wiederholt werden;
- Definierte Parameter: Definition eigener Variablen, die nicht unbedingt mit dem Netzwerk zusammenhängen müssen. Da auch die zeitliche Ableitung dieser Variablen angegeben werden kann, ist die direkte Integration von Differentialgleichungssystemen ohne Netzwerkbeschreibung möglich;

- Ausgabe der Ergebnisse: tabellarisch oder als Druckerplots; Darstellung mehrerer Größen in einem Diagramm, nicht nur in Abhängigkeit von der Zeit; Schnittstelle für externe Grafikprogramme vorhanden[1];
- linear abhängige Quellen: Spannungen und Ströme, die linear von Strömen durch Widerstände und Induktivitäten und/oder Spannungen an Widerständen und Kapazitäten[2] abhängen, stehen ohne Fehler (Zeitverzug, *computational delay*) zur Verfügung;
- automatischer Abbruch einer Simulation in Abhängigkeit vom Netzwerkverhalten.

Die Verwendung definierter Parameter und ihrer zeitlichen Ableitungen sind eine der herausragenden Merkmale von SCEPTRE. Damit können Differentialgleichungssysteme direkt gelöst werden. Mit dieser Möglichkeit erschließt sich dem Anwender auch die Simulation nicht-elektrischer und interdisziplinärer Probleme. Weiterhin können mehrere Schaltungsteile simuliert werden, die nicht galvanisch miteinander verbunden sein müssen (es gibt kein gemeinsames Referenzpotential). Damit lässt sich das Verhalten von zwei oder mehr Schaltungen in einem Lauf miteinander vergleichen. SCEPTER besitzt eine Schnittstelle zu eigenen FORTRAN-Unterprogrammen. Diese und andere Eigenschaften erhöhen die Universalität von SCEPTRE gegenüber anderen Simulationsprogrammen [4].

1.2 Hard- und Softwarevoraussetzungen für Linux/UNIX

- Linux/UNIX Betriebssystem;
- FORTRAN-Compiler, z. B. **gfortran**
- Grafikpaket **gnuplot** für das Auswerteprogramm NGP (siehe Kapitel 7.5).

1.3 Entwicklungsgeschichte

SCEPTRE wurde 1966 im Auftrag der US-Amerikanischen Regierung von der Firma IBM entwickelt. Ursprünglich war dieses Programm für die Rechnerfamilie IBM 7090/94 konzipiert. Jahre später stellte AFWL (Air Force Weapons Laboratory, AFB, New Mexico, USA) das Programm kostenlos für die Rechnersysteme IBM S/360 und CDC 6000/7000 zur Verfügung. Um die Portabilität zwischen Rechnersystemen zu wahren wurde SCEPTRE vollständig in FORTRAN IV programmiert. In der S/360-Version wurden Teile der Ein-/Ausgabeoperationen sowie Routinen zu den Bit- und Byte-Manipulationen in Assembler geschrieben (siehe Kapitel C.2). Eine Weiterentwicklung von SCEPTRE wurde von IBM als Lizenzprogramm unter dem Namen ASTAP angeboten [5].

1.4 Beispiel

Das folgende Beispiel zeigt die Vorgehensweise, um die Schaltung nach Abb. 1.1 zu simulieren. Es sei angenommen, dass das Ersatzschaltbild für den Transistor (2N914A) bereits in der Modellbibliothek gespeichert ist. Die für SCEPTRE aufbereitete Schaltung zeigt Abb. 1.2. Nach der Wahl eines geeigneten Maßeinheitensystems lässt sich das Netzwerk durch folgende SCEPTRE-Anweisungen beschreiben:

[1] Die vorliegende Linux-Version schreibt die Ergebnisse auf eine direkte Zugriffsdatei, die man mit einem separaten Programm (NGP) und dem Open-Source Programm **gnuplot** auswerten kann.
[2] Zustandsvariablen

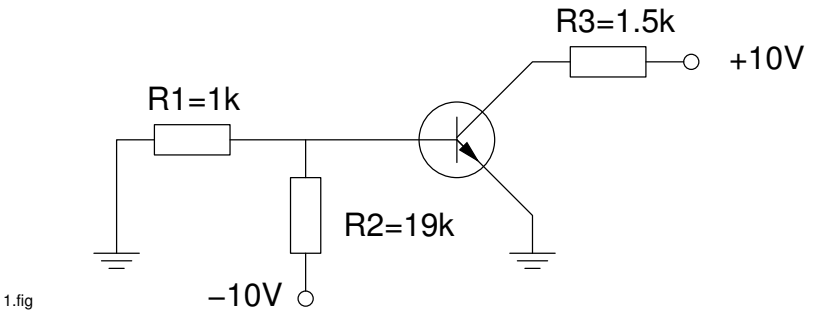


Abbildung 1.1: Einfache Inverter-Schaltung

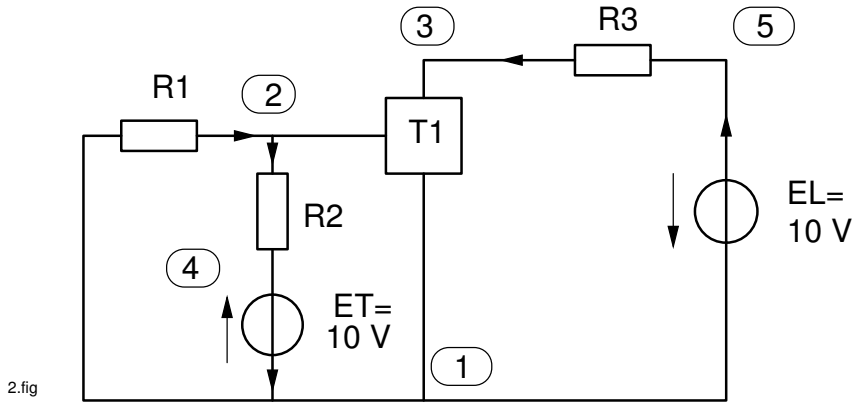


Abbildung 1.2: SCEPTRE Darstellung der Inverter-Schaltung

```
CIRCUIT DESCRIPTION
ELEMENTS
Rl,1-2 = 1
R2,2-4 = 19
R3,5-3 = 1.5
EL,1-5 = 10
ET,4-1 = 10
Tl,2-1-3 = MODEL 2N914A (PERM)
OUTPUTS
VR3, VR1, VCXT1, PLOT
RUN CONTROLS
STOP TIME = 500, RUN INITIAL CONDITIONS
END
```

Zuerst wird der Arbeitspunkt der Schaltung bestimmt (`RUN INITIAL CONDITIONS`) und mit diesen Werten das transiente Verhalten der Schaltung für 500 ns simuliert (`STOP TIME = 500`). Ausgegeben werden sowohl tabellarisch als auch als Druckerplot die Verläufe der Spannungen an den Widerständen R_1 und R_3 (`VR3, VR1`) als auch am Kondensator C_X, der zu dem Ersatzschaltbild des Transistors T_1 gehört (`VCXT1`).

Man beachte, dass die Wertzuweisung zu den physikalischen Größen (Spannung, Strom, Widerstand, Kapazität, Zeit, …) ohne Angabe von Maßeinheiten geschieht. Es gibt auch keine Voreinstellungen, sondern der Benutzer sucht sich ein bequemes aber konsistentes Maßeinheitensystem selber aus (siehe dazu Abschnitt 2.2 und Tabelle 2.3, Seite 9).

1.5 Hinweise zu diesem Handbuch

Dieses Handbuch lehnt sich an die englische Originaldokumentation [2, 3] an, ist aber nicht als vollständige Übersetzung anzusehen. An einigen Stellen wurden zusätzliche Informationen hinzugefügt, die auf Erfahrungen des Autors beruhen, an anderen Stellen wurden Passagen weggelassen bzw. auf die Originaldokumentation verwiesen. Zusätzlich wurde ein Kapitel über die Benutzung von SCEPTRE unter Linux/UNIX eingefügt.

Rechnerspezifische Angaben wurden teilweise mit in die Dokumentation übernommen, aus historischen Gründen. Immerhin sind seit der Entwicklung von SCEPTRE etwa 50 Jahre vergangen, in denen die Entwicklung der Rechnerhardware mehrere Quantensprünge vollzogen hat.

In Anlehnung an die englischen Schreibweisen werden in diesem Handbuch Dezimalzahlen mit Dezimalpunkt statt mit Dezimalkomma geschrieben.

Teil I

Programmbeschreibung (Volume I)

2 SCEPTRE-Anweisungen

2.1 Sprachvereinbarungen

Die allgemeine Form einer SCEPTRE-Anweisung wird in diesem Handbuch eingerahmt dargestellt. Alle Schlüsselwörter müssen in Großbuchstaben geschrieben werden, z. B.

`RUN INITIAL CONDITIONS ONLY` .

Vom Benutzer vorzugebene Parameter werden kursiv gedruckt, z. B.

`M` *name*, `L` *name1* – `L` *name2* .

Geschweifte Klammern deuten alternative Eingaben an:

$$\left\{ \begin{array}{l} \texttt{DE} \\ \texttt{DJ} \end{array} \right\} \textit{name} = \textit{value} \quad .$$

Wahlweise Angaben werden durch eckige Klammern gekennzeichnet:

variable1 [, *variable2*...], `PLOT (`*variable3*`)` .

Voreinstellungen sind entweder unterstrichen oder stehen neben der Umrahmung.

An verschiedenen Stellen taucht der Begriff 'Karte' auf. Er ist gleichbedeutend mit der Zeile einer Datei.

Zur Vereinfachung werden in diesem Handbuch nachstehend aufgeführte Vereinbarungen getroffen. Die englische Bezeichnung wurde im allgemeinen beibehalten:

element name: Name einer Netzwerkkomponente
Beispiel: `R102, CLE, E1, LCOIL`

node: Name eines Knotenpunktes
Beispiel: `0, GND, MASSE, EGON`

number: numerische Konstante im D-, E-, I- oder F-Format, mit oder ohne Vorzeichen, max. 13 Zeichen lang
Beispiel: `314, 3.14, -.314, 3.14E-3`

constant: wie **number**, jedoch immer mit Dezimalpunkt

value: dazu gehören:

- **number**
- **defined parameters**
- **TABLE...**

- **EQUATION…**
- **EXPRESSION…**
- **external function**

special value: dazu gehören:

- **value**
- **constant** `*` `IR…`, **constant** `*` `VR…`
  Beispiel: `3.*IRL, .5E-7*VR2`
- **value** `*` `J…`
  Beispiel: `PS3*JEIN, T43(VLS)*JAUS`
- **DIODE TABLE…**
- **DIODE EQUATION…**

variable: dazu gehören:

- Spannungen und Ströme (**V…, I…**)
  Beispiel: `VRL, IRL, VJEIN, IEIN`
- Quellen und ihre Ableitungen (**E…, DE…, J…, DJ…**)
  Beispiel: `E1, DE1, J2, DJ2`
- **defined parameters**
- Wert einer Netzwerkkomponente (**element name**)
  Beispiel: `R1, CES, ML1L2`
- **TIME**
- **interne SCEPTRE-Parameter**
  Beispiel: `XSTPSZ, XIR, XTMON`

TABLE *name(variable)*: Netzwerkgrößen können auch in tabellarischer Form vorliegen. Die Tabelle wird unter `FUNCTIONS` definiert. *Variable* entspricht der unabhängigen Variablen. Fehlt die Variable, wird **TIME** als unabhängige Variable angenommen.
Verkürzte Schreibweise: `T`*name(variable)*
Beispiele:
`TABLE UIC(VC1)`
`TREGEL(PU4)`[1]

EQUATION *name(arg1, arg2,…)*: Werden Netzwerkgrößen nach einer arithmetischen Anweisung berechnet, kann man diese Schreibweise benutzen. Sie hat die Wirkung eines Funktionsaufrufs, bei dem die Parameter *arg1, arg2,…* übergeben werden. Die mathematische Formulierung wird unter `FUNCTIONS` beschrieben. Die Argumente können vom Typ **variable**, **constant**, bzw. **TABLE…** sein. Die **EQUATION**-Anweisung ist dann vorteilhaft, wenn mehrere Netzwerkgrößen nach der gleichen Vorschrift mit verschiedenen Zahlenwerten berechnet werden sollen.
Verkürzte Schreibweise: `Q`*name (arg1, arg2,…)*
Beispiele:
`EQUATION A1 (E1, TIME,23.4,TABLE9)`
`QA1(220.,TIME,RA,T9(VRA))`

[1] siehe auch Abschnitt 3.4

EXPRESSION *name (arithmetische Anweisung)*: Diese Schreibweise hat die gleiche Wirkung wie **EQUATION**. Sie erlaubt jedoch die direkte Angabe der Berechnungsvorschrift und erweist sich als vorteilhaft beim einmaligen Berechnen von Netzwerkgrößen. Um Verwechselungen mit internen SCEPTRE-Parametern zu vermeiden, sollte *name* möglichst nur aus Ziffern bestehen. Die mathematische Formulierung ist identisch mit der in FORTRAN. Neben den 5 Grundrechenarten (`+, -, /, *, **`) können auch **external function**s benutzt werden. Numerische Konstanten müssen mit einem Dezimalpunkt angegeben werden.
Verkürzte Schreibweise: `X`*name(…)*
Beispiele:
```
E1=X100(310.*DSIN(314.*TIME))
POM=X209(2.*PI*TIME)
PI=X210(4.*DATAN(1.))
CBE=X2(IR5*XTABLE(T23,VRA))
```
[2]

DIODE TABLE *name (variable)*: Liegt eine Diodenkennlinie in Tabellenform vor, so bezieht sich *variable* automatisch auf die Diodenspannung
Beispiel: `JA=DIODE TABLE OY100 (VCJA)`

DIODE EQUATION *(x_1, x_2)*: Eine Diodenkennlinie kann auch in der Form $J = x_1(e^{x_2 V_J} - 1)$ vorliegen. Die Diode wird automatisch durch eine nach der o. a. Formel gesteuerten Stromquelle `J` ersetzt, `VJ` ist die Diodenspannung.
Beispiel: `DIODE EQUATION (3.E-14,38.)`

external function: Für arithmetische Anweisungen stehen alle Funktionen der FORTRAN Unterprogrammbibliothek sowie selbstgeschriebene FORTRAN Unterprogramme zur Verfügung. Es müssen die doppeltgenauen Aufrufe benutzt werden, z.B. für reelle Argumente `DSIN, DCOS, DEXP, DATAN, DLOG10`…. Für komplexe Argumente (AC-Analyse) stehen folgende Funktionen in etwas abgeänderter Schreibweise zur Verfügung: `ZSQRT, ZSIN, ZCOS, DZABS, ZLOG, ZCMPLX`.

interne SCEPTRE-Parameter: Diese Programmvariablen werden von SCEPTRE benötigt; sie stehen dem Benutzer in den Eingabeanweisungen direkt und in selbstgeschriebenen FORTRAN Unterprogrammen über einen COMMON-Bereich zur Verfügung.
Beispiele: `XSTPSZ, XIR, XTMON, XSTOPT`.

2.2 Vorbereitungen

Zuerst wird das Ersatzschaltbild des zu untersuchenden Netzwerks gezeichnet und jedem Element ein Name zugeordnet. Dabei klassifiziert der 1. Buchstabe des Namens das Element gemäß Tabelle 2.1.

Weiterhin werden alle Knotenpunkte mit einem Namen versehen. Ein Name darf nur aus alphanumerischen Zeichen bestehen. Die Länge der frei wählbaren Namen ist begrenzt (siehe Tabelle 2.2).

Die mit ‡ markierten Namen dürfen auch mit numerischen Zeichen beginnen (z.B. `2N3055` als Modellname oder `0` als Knotenpunktname). Die mit † versehenen Zahlenangaben sind empfohlene Werte und sollten möglichst nicht überschritten werden.

Die Maßeinheiten können frei gewählt werden. Sie müssen nur die Verknüpfungsgleichungen

$$U = IR \qquad \tau = \frac{L}{R} = RC \qquad f = \frac{1}{T}$$

[2] wegen der geänderten Syntax siehe Abschnitt 3.4

Tabelle 2.1: Klassifizierung der Elemente

1. Buchstabe	Element
R	Widerstand
C	Kapazität
L	Induktivität
M	Gegeninduktivität
E	Spannungsquelle
J	Stromquelle

Tabelle 2.2: Länge der frei wählbaren Namen

	circuit description	model description
Knotenpunkte ‡	6	3 †
Elemente	5	2 †
Definierte Parameter	6	3 †
Tabellennamen	5	2 †
EQUATION-Namen	5	2 †
Modellnamen ‡	18	18
Ausgabevariablen	6	3 †
Modellaufrufe	3 †	-

widerspruchsfrei erfüllen (siehe Tabelle 2.3).

Tabelle 2.3: Beispiele für konsistente Maßeinheitensysteme

Widerstand	Ω	kΩ	kΩ	kΩ
Kapazität	F	pF	μF	nF
Induktivität	H	μH	H	mH
Strom	A	mA	A	mA
Spannung	V	V	kV	V
Zeit	s	ns	ms	μs
Frequenz	Hz	GHz	kHz	MHz

Außer den Netzwerkelementen können in SCEPTRE auch Tabellen, arithmetische Anweisungen und andere Größen definiert werden. Sie werden ebenfalls durch ihren 1. Buchstaben gekennzeichnet entsprechend Tabelle 2.4.

In jeden Stromzweig legt man eine Stromrichtung, für die Spannungsquellen die Polarität fest. Dabei ist zu beachten, daß SCEPTRE bei Verbrauchern Spannungs- und Strompfeile gleich-sinnig, bei Quellen gegen-sinnig annimmt.

Tabelle 2.4: Klassifizierung weiterer SCEPTRE-Größen

1.Buchstabe	Bedeutung
`P`	definierter Parameter, reell
`W`	definierter Parameter, komplex
`T, TABLE`	Tabelle
`Q, EQUATION`	arithmetische Anweisung als Formelaufruf
`X, EXPRESSION`	arithmetische Anweisung, direkt
`V`	Spannung an einem Netzwerkelement
`I`	Strom durch ein Netzwerkelement
`D`	Differential $\mathrm{d}/\mathrm{d}\,t$
`G`	vollständiges Differential
`F`	eigene FORTRAN-Unterprogramme

2.3 Eingabeformat

Alle SCEPTRE-Anweisungen müssen in Großbuchstaben zwischen den Spaltenpositionen 1 und 72 einer Textdatei geschrieben werden. Die Eingabe ist formatfrei, Leerzeichen und -zeilen können zur besseren Lesbarkeit eingefügt werden. Mehrere Anweisungen dürfen, durch Kommata getrennt, in einer Zeile geschrieben werden. Anweisungen können auf mehreren Eingabezeilen fortgesetzt werden, und zwar unmittelbar nach den Zeichen

```
+  -  *  /  ,  .  (  )  =.
```

SCEPTRE ignoriert innerhalb einer Zeile alle Informationen nach einem Semikolon ";".

Die Datei darf keine Sonderzeichen, z. B. TAB, enthalten.

2.4 Eingabestruktur

Tabelle 2.5 zeigt die Struktur der SCEPTRE-Eingabe. Die Reihenfolge der Anweisungen innerhalb einer Anweisungsebene ist beliebig. Zu jeder der o. a. Steueranweisungen gehören Daten, die nachfolgend im einzelnen beschrieben werden. Fehlen Daten zu einer Steueranweisung, kann diese ebenfalls entfallen.

2.5 ELEMENTS

Alle Netzwerkkomponenten werden unter dieser Überschrift definiert, und zwar in der allgemeinen Form:

element, node1 – node2 = value

Dabei wird festgelegt, dass der Strom durch *element* von *node1* nach *node2* fließt.

Tabelle 2.5: Struktur der SCEPTRE Eingabedaten

Hauptüberschrift	1. Anweisungsebene	2. Anweisungsebene
CIRCUIT DESCRIPTION	maximal 11 Kommentarkarten	
	ELEMENTS	
	DEFINED PARAMETERS	
	OUTPUTS	
	FUNCTIONS	
	INITIAL CONDITIONS	
	RUN CONTROLS	SENSITIVITY
		MONTE CARLO
		WORST CASE
		OPTIMIZATION
MODEL DESCRIPTION	MODEL name	maximal 11 Kommentarkarten
		ELEMENTS
		DEFINED PARAMETERS
		OUTPUTS
		FUNCTIONS
		INITIAL CONDITIONS
RERUN DESCRIPTION	maximal 11 Kommentarkarten	
	ELEMENTS	
	DEFINED PARAMETERS	
	INITIAL CONDITIONS	
	FUNCTIONS	
	RUN CONTROLS	
CONTINUE	RUN CONTROLS	
RE-OUTPUT	OUTPUTS	
	RUN CONTROLS	
END		

2.5.1 Allgemeine Zweipolform

Für die entsprechenden Netzwerkelemente lautet die Eintragung (siehe auch Tabelle 2.6) :

$$\left\{\begin{array}{l}\texttt{R}\\ \texttt{C}\\ \texttt{L}\\ \texttt{E}\\ \texttt{J}\end{array}\right\}\ \textit{name, node1} - \textit{node2} = \textit{value}$$

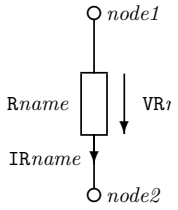
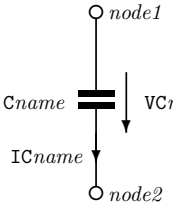
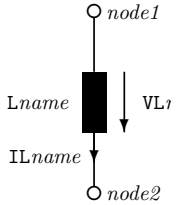
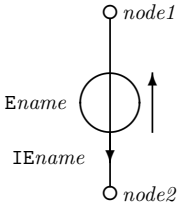
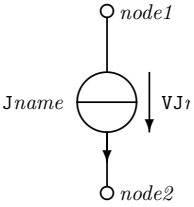
	Rname , node1 - node2 = value
	Cname , node1 - node2 = value
	Lname , node1 - node2 = value
	Ename , node1 - node2 = value
	Jname , node1 - node2 = value

Tabelle 2.6: Festlegung von Strom und Spannung bei allgemeinen Zweipolen

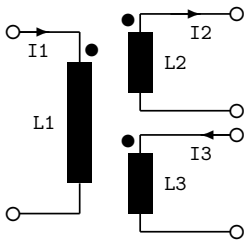
2.5.2 Gegeninduktivitäten

`M` *name*, `L` *name1* – `L` *name2* = *value*

`L`*name1* und `L`*name2* sind die beiden miteinander gekoppelten Induktivitäten. Es muß die Bedingung $M < \sqrt{L_1 L_2}$ erfüllt sein.

Das Vorzeichen von `M` ist positiv, wenn die induzierten Spannungen der Gegen- und Selbstinduktivitäten in einer Wicklungsanordnung die gleichen Richtungen haben. Für das dargestellte Beispiel gilt:

```
M12, L1 - L2 = - value
M13, L1 - L3 = + value
M23, L2 - L3 = - value
```



2.5.3 Widerstandsabhängige Quellen

`E` *name, node1 – node 2 = constant* `* VR…`

`J` *name, node1 – node 2 = constant* `* IR…`

2.5.4 Gesteuerte Stromquellen

Gesteuerte Stromquellen dienen vorwiegend zur Simulation von Halbleiterschaltungen.

Stromquellen 1. Ordnung

`J` *name, node1 – node2* = { `DIODE TABLE…` | `DIODE EQUATION…` }

Stromquellen 2. Ordnung

`J` *name1, node1 – node2 = value* `* J` *name2*

`J`*name2* ist eine Stromquelle 1. Ordnung.

2.5.5 Zeitliche Ableitungen von Quellen

Treten in einem Netzwerk Maschen auf, die nur eine Spannungsquelle und einen Kondensator bzw. eine Stromquelle und eine Induktivität enthalten, verlangt SCEPTRE zusätzlich die zeitliche Ableitung dieser Quellen[3].

$$\left\{ \begin{array}{l} \texttt{DE} \\ \texttt{DJ} \end{array} \right\} \textit{name} = \textit{value}$$

2.5.6 Modellaufruf

name, node1 – node2 – … `= MODEL` *modelname*

2.5.7 Elemente mit Toleranzgrenzen

Diese Angaben werden nur bei `MONTE CARLO`, `WORST CASE` und/oder `OPTIMIZATION` benötigt.

$$\left\{ \begin{array}{l} \texttt{R} \\ \texttt{E} \\ \texttt{J} \end{array} \right\} \textit{name, node1 – node2} = \textit{number1} \left\{ \begin{array}{l} (\textit{number2, number3}) \\ (\textit{number4}) \end{array} \right\}$$

number1 entspricht dem Nominalwert, *number2* dem unteren, *number3* dem oberen Wert, *number4* gibt die prozentuale Abweichung an.

2.5.8 Quellen bei der Wechselstromanalyse

$$\left\{ \begin{array}{l} \texttt{E} \\ \texttt{J} \end{array} \right\} \textit{name, node1 – node2} = (\left\{ \begin{array}{l} \textit{constant} \\ \textit{def. par.} \\ \texttt{TABLE…} \\ \texttt{FREQ} \end{array} \right\}, \left\{ \begin{array}{l} \textit{constant} \\ \textit{def. par.} \\ \texttt{TABLE…} \\ \texttt{FREQ} \end{array} \right\}) , \left\{ \begin{array}{l} \underline{\texttt{DEGREES}} \\ \texttt{RADIANS} \\ \texttt{COMPLEX} \end{array} \right\}$$

Bei der Angabe `DEGREES` (Phasenwinkel in Grad) oder `RADIANS` (Phasenwinkel im Bogenmaß) stellt das erste Argument die Amplitude, das zweite den Phasenwinkel dar. Benutzt man `COMPLEX`, dann enthält das erste Argument den Realteil, das zweite den Imaginärteil.

Alle R-, C-, L- und M-Werte werden bei einer Wechselstromanalyse als konstant angenommen. SCEPTRE setzt automatisch die vom Benutzer vorgegebenen Anfangsbedingungen ein oder, falls diese fehlen, berechnet sie zum Zeitpunkt $t = 0$.

Alle konstanten und zeitabhängigen Quellen werden bei der Wechselstromanalyse durch den Wert Null ersetzt.

Bei der Wechselstromanalyse ist im Gegensatz zur transienten Analyse unter bestimmten netzwerktopologischen Voraussetzungen die Angabe der zeitlichen Ableitungen einer Quelle nicht erforderlich. Sie ergibt sich automatisch durch Multiplikation mit $\mathrm{j}\,\omega$.

[3] Diesen Fall kann man vermeiden, indem man einen geeigneten Widerstand in diese Maschen einfügt.

2.6 DEFINED PARAMETERS

Unter diesem Titel können zusätzliche Zwischenvariablen definiert werden, z.B. um aus mehreren Netzwerkgrößen eine neue Größe zu berechnen. Allgemeine Form:

`P`*name* = *value*

Beispiele:

```
PR5 = X10(IR5 * VR5)
PTAU = X11(C1 * R9)
```

Die zeitliche Ableitung einer Variablen $\frac{\mathrm{d}}{\mathrm{d}\,t}(P(t))$ kann ebenfalls angegeben werden:

`DP`*name* = *value*

Beispiel zur Lösung eines Differentialgleichungssystems 1. Ordnung:

$$\begin{aligned} \dot{x} = &\ -6x & +5y & & +10 \\ \dot{y} = &\ 5x & -7y & +0,2z & \\ \dot{z} = &\ & 0,2y & -0,2z & -0,5 \end{aligned}$$

Anfangsbedingungen: $x(0) = 6, y(0) = 5, z(0) = 4$

```
DPX=X1(-6.*PX+5.*PY+10.)
DPY=X2(5.*PX-7.*PY+0.2*PZ)
DPZ=X3(0.2*PY-0.2*PZ-0.5)
PX=6
PY=5
PZ=4
```

Zwischenvariablen können auch Toleranzgrenzen haben[4]:

`P` *name* = *number1* { `(`*number2, number3*`)` / `(`*number4*`)` }

Das vollständige Differential der Funktion $P(x, y, ...)$, das bei SENSITIVITY, OPTIMIZATION und WORST CASE benötigt wird,

$$\mathrm{d}\,P = \frac{\partial P}{\partial x}dx + \frac{\partial P}{\partial y}dy + ...$$

wird in SCEPTRE folgendermaßen definiert:

[4] siehe Kapitel 2.5: Elemente mit Toleranzgrenzen

`GP`*name* `=` `P`*name1* `*` `D`*name2* `+` `P`*name3* `*` `D`*name4* `+ …`

Dabei stellen `P`*name1*, `P`*name3* die partiellen Ableitungen $\partial P/\partial x, \partial P/\partial y$, `D`*name2*, `D`*name4* die Differentiale $\mathrm{d}\,x, \mathrm{d}\,y$ dar. Das vollständige Differential *muss* in dieser Form geschrieben werden (keine `EXPRESSION`- oder `EQUATION`-Anweisung).

Beispiele:

```
PWR = X1(IRL ** 2. + ILX ** 2.)
GPWR= P2 * DIRL + P3 * DILX
P2  = X76(2. * IRL)
P3  = X77(2. * ILX)
```

Die Differentiale `D`*name2*, `D`*name4* …müssen angegeben werden, wenn die Variablen *name2*, *name4* …als unabhängige Variablen definiert werden.

Für die AC-Analyse müssen die Zwischenvariablen komplex sein. Ihnen kann man reelle Werte

`W`*name* `=` *value*

oder komplexe Werte

`W`*name* `=` `(`*value1*`,` *value2*`)` *typ*

zuordnen[5]. Komplexe Zwischenvariablen können nur zur Berechnung von Größen benutzt werden, die ausgegeben werden sollen, d.h. sie dürfen weder in einer EQUATION-, EXPRESSION- oder TABLE-Anweisung noch in einer Funktion auftreten.

2.7 OUTPUTS

Unter diesem Titel können alle Größen vom Typ **variable** aufgeführt werden. SCEPTRE bietet die Möglichkeit, die Ergebnisse sowohl in Abhängigkeit von der Zeit/Frequenz als auch von anderen unabhängigen Variablen auszudrucken/plotten. Allgemein erhält man einen tabellarischen Ausdruck in Abhängigkeit von der Zeit/Frequenz mit der Anweisung

variable1 [, *variable2…*] .

Einen zusätzlichen Schnelldruckerplot, und zwar jede Variable einzeln auf einer Seite, erhält man mit

variable1 [, *variable2…*], `PLOT` .

Plotterzeichnungen als Funktionen anderer unabhängiger Variablen werden angefordert mit

[5] Bedeutung der Parameter siehe Kapitel 2.5.8: Quellen bei der Wechselstromanalyse

variable1 [, *variable2*...], `PLOT (`*variable3*`)` .

Außer `TIME` können alle Größen umbenannt werden mit

variable1 `(`*name1*`)` [, *variable2* `(`*name2*`)` ...], `PLOT (`*variable3* `(`*name3*`))` .

Die neuen Namen *name1*... dürfen maximal 6 Zeichen lang sein.
Beispiele:

```
E1, IE1, VRL, XSTPSZ, PLOT
E1(U1),IE1(I1),VRL(U2),IRL(I2),PLOT
VRL(U2), PLOT (IRL(I2))
```

In den o. a. Beispielen wird jede Variable einzeln auf eine Schnelldruckerseite geplottet. Bis zu 9 Größen können aber auch gleichzeitig über mehrere Seiten hinweg in ein Diagramm gezeichnet werden, jedoch nur in Abhängigkeit von `TIME`, und zwar indem man der Plotterzeichnung einen Namen gibt (max. 6 alphanumerische Zeichen):

variable1 [, *variable2*...], `PLOT` *plotname* .

Es können mehrere Zeichnungen dieser Art durch verschiedene Plotternamen angefordert werden. Eine Umbenennung der Ausgabevariablen ist ebenfalls möglich. Da die Länge einer solchen Zeichnung bis zu 2000 Zeilen betragen kann, muss unter RUN CONTROLS die Anweisung

`PLOT INTERVAL =` *number*

eingefügt werden[6].

Die Ergebnisse einer AC-Analyse werden normalerweise als Funktion der Frequenz ausgegeben.

variable1 [, *variable2*...] , { DEGREES | RADIANS | COMPLEX | NYQUIST } [,PLOT]

Mit `NYQUIST` wird der Imaginärteil über dem Realteil aufgetragen, das Wort `PLOT` kann dabei entfallen. Mit `DEGREES` erhält man die Amplitude und den Phasenwinkel in Grad, mit `RADIANS` den Phasenwinkel im Bogenmaß und mit `COMPLEX` den Real- und Imaginärteil der entsprechenden Wechselstromgröße.

[6] siehe Kapitel 2.10.7

2.8 INITIAL CONDITIONS

Für Kapazitäten, Induktivitäten und Dioden-Stromquellen können Anfangsbedingungen gesetzt werden:

$$\left\{ \begin{array}{c} \texttt{VC} \\ \texttt{IL} \\ \texttt{VJ} \end{array} \right\} \textit{name} = \textit{number} \quad .$$

SCEPTRE bietet zwei Möglichkeiten, die Anfangsbedingungn für eine Netzwerksimulation anzugeben:

- Aufruf der Gleichstromanalyse (`RUN IC`). In diesem Fall werden die Anfangsbedingungen vom Programm berechnet. Die o. a. Anweisungen dienen dann nur als Anfangswerte für die Newton-Raphson-Iteration. Andernfalls beginnt die Iteration bei Null.
- Direkte Eingabe der Anfangswerte. Fehlen diese, werden alle Anfangswerte automatisch auf den Wert Null gesetzt.

2.9 FUNCTIONS

Unter dieser Überschrift werden alle Tabellen und Funktionen, die unter `ELEMENTS` und `DEFINED PARAMETERS` benötigt werden, beschrieben.

2.9.1 EQUATION-Anweisung

$$\left\{ \begin{array}{c} \texttt{EQUATION} \\ \texttt{Q} \end{array} \right\} \textit{name}\ (\textit{Liste}) = (\textit{arithmetische Anweisung})$$

Liste enthält Hilfsvariablen (Formalparameter), deren Namen maximal 6 Zeichen lang sein und nicht mit einer Zahl oder den Buchstaben `I` bis `N` beginnen dürfen[7]. Diese Hilfsvariablen werden in einer arithmetischen Anweisung entsprechend der in FORTRAN üblichen Formelfunktion miteinander verknüpft. Es können auch externe Unterprogramme aufgerufen werden. Alle numerischen Konstanten in der arithmetischen Anweisung müssen mit Dezimalpunkt geschrieben werden. Wird mit komplexen Größen gerechnet, müssen die Hilfsvariablen mit dem Buchstaben `Z` beginnen. Die Anzahl der Hilfsvariablen muss mit der Anzahl der Parameter im `EQUATION`-Aufruf übereinstimmen. Die Namen der Hilfsvariablen können in anderen `EQUATION`-Anweisungen wiederbenutzt werden.

Beispiele:

```
ELEMENTS
  E1, 1-0 = Q9 (220.,50.)
  ...
FUNCTIONS
  Q9 (A,B) = (A*DSQRT(2.) * DSIN (B*6.28 * TIME))
```

[7] wegen der Wahl der Variablennamen siehe Abschnitt 3.19, Seite 53

oder

```
DEFINED PARAMETERS
  WZE = Q107 (EIN,IEIN)
  ...
FUNCTIONS
  Q107 (Z1,ZB7X) = (Z1/ZB7X)
```

2.9.2 TABLE-Anweisung

[DIODE] { TABLE | T } *name* { = | , | neue Zeile } *number1 , number2 , ...*

Die Tabellenwerte werden paarweise, durch Kommata getrennt, eingegeben. Der erste Wert entspricht der unabhängigen, der zweite der abhängigen Variablen. Die Werte der unabhängigen Variablen müssen in monoton aufsteigender Reihenfolge vorliegen. Die Funktion muss eindeutig sein. Es können aber zu einem unabhängigen Wert zwei abhängige Werte angegeben werden, um Sprungfunktionen zu beschreiben (siehe Beispiel). Zwischenwerte in den Tabellen werden linear inter- oder extrapoliert[8]. Beispiel: Beschreibung einer Spannungsfunktion nach Abbildung 2.1.

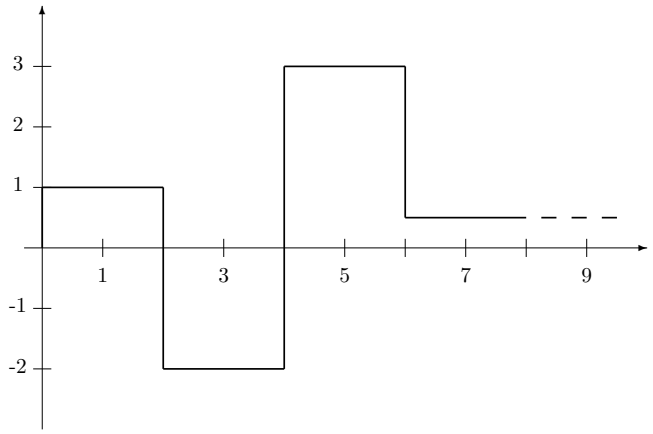


Abbildung 2.1: Spannungsverlauf

```
TABLE XYZ7
0,0, 0,1, 2,1, 2,-2, 4,-2, 4,3,
6,3, 6,0.5, 100,0.5
```

Wird die Tabelle als `DIODE TABLE` definiert, nimmt SCEPTRE an, dass der erste Wert der Diodenspannung, der zweite dem entsprechenden Diodenstrom entspricht.

[8] besonders wichtig für das letzte Wertepaar

2.10 RUN CONTROLS

Zu diesem Kapitel gehören eine Reihe von Steueranweisungen, mit denen sich der Rechenablauf einer Simulation in weiten Grenzen steuern lässt. Es stehen drei Analysearten zur Verfügung, die teilweise miteinander kombiniert werden können.

2.10.1 DC-Analyse

Soll nur eine Gleichstromanalyse durchgeführt werden, genügt die Anweisung

`RUN INITIAL CONDITIONS ONLY` .

In diesem Fall wird die DC-Analyse mit dem Newton-Raphson-Algorithmus durchgeführt. Die maximale Anzahl der Iterationen kann zusätzlich beeinflusst werden mit

`NEWTON-RAPHSON PASS LIMIT =` *number* (100).

Die Konvergenzkriterien für die Iteration können ebenfalls geändert werden durch die Anweisungen

`RELATIVE CONVERGENCE =` *number* (10^{-3})

`ABSOLUTE CONVERGENCE =` *number* (10^{-4}) .

Mit

`RUN IC VIA IMPLICIT`

wird anstatt des Newton-Raphson-Verfahrens ein impliziter Algorithmus aufgerufen.

Innerhalb der DC-Analyse stehen vier spezielle Verfahren zur Verfügung, die neben den Steuerkarten unter `RUN CONTROLS` zusätzlich noch Anweisungen unter `CIRCUIT DESCRIPTION` benötigen. Für MONTE-CARLO, OPTIMIZATION und WORST-CASE muss unter `ELEMENTS` bzw. `DEFINED PARAMETERS` mindestens eine Größe mit Toleranzgrenzen angegeben werden.

Unabhängig von der Reihenfolge der Steueranweisungen werden – falls verlangt – die Toleranzanalysen in der Reihenfolge

1. SENSITIVITY,
2. MONTE-CARLO,
3. WORST-CASE und
4. OPTIMIZATION

durchgeführt.

Sensitivity:

Es genügt die Anweisung

`RUN SENSITIVITY` .

Monte Carlo:

Der Aufruf lautet

`RUN MONTE CARLO [ = ` *number* `]` (10) .

Mit *number* kann die Anzahl der Iterationen variiert werden. Die Verteilung der Zufallszahlen wird gesteuert mit

`DISTRIBUTION =` { `GAUSSIAN` | `UNIFORM` } .

Man kann die Anfangszufallszahl vorgeben mit

`INITIAL RANDOM NUMBER =` *number* (127263527) .

Diese Zahl sollte positiv, ungerade und neun Stellen lang sein (I-Format). Soll bei Wiederholungsläufen immer mit der gleichen Zufallszahl begonnen werden, erreicht man dies durch

`INITIAL RANDOM NUMBER = DEFAULT` .

Zusätzliche Informationen zu jeder Monte-Carlo-Iteration erhält man mit

`LIST MONTE CARLO DETAILS` .

Worst-Case:

Eine Worst-Case-Analyse erhält man mit

`RUN WORST CASE` [`=` { `LOW` | `NOMINAL` | `HIGH` }] .

Die Schlüsselwörter nach dem Gleichheitszeichen geben an, welche Werte als Anfangswerte für eine nachfolgende TR-Analyse eingesetzt werden sollen.

Optimization:

Diese Analyse wird mit

`RUN OPTIMIZATION [ = ` *number* `]` (30)

aufgerufen. Mit *number* kann die Anzahl der durchzuführenden Iterationen geändert werden. Zwischenergebnisse erhält man mit

`LIST OPTIMIZATION DETAILS` .

Mit den folgenden Steueranweisungen kann die Optimierung in weiten Grenzen beeinflußt werden:

`INITIAL H MATRIX FACTOR =` *number* (1.)

`OPTIMIZATION CRITERION =` *number* (10^{-7})

`OPTIMIZATION RANDOM STEPS =` *number* (0)

`RANDOM STEP SIZE CONTROL =` *number* (0.2)

`MINIMUM FUNCTION ESTIMATE =` *number* (0)

`PUNCH OPTIMIZATION RESULTS` .

Die Bedeutung und Handhabung dieser Anweisungen sind in [3] beschrieben.

2.10.2 AC-Analyse

Die AC-Analyse wird aufgerufen durch die Anweisung

`RUN AC` .

Soll das Netzwerk nur für eine Frequenz berechnet werden, genügt die Anweisung

`FREQUENCY =` *number* .

Für die Beschreibung eines Frequenzbereichs werden folgende Angaben benötigt:

`INITIAL FREQUENCY =` *number*

`FINAL FREQUENCY =` *number*

`NUMBER FREQUENCY STEPS =` *number* (10)

`TYPE FREQUENCY RUN =` { LINEAR | LOG } .

Werden bei Wiederholungsläufen nur die Strom- und/oder Spannungsquellen variiert, spart man Rechenzeit mit der Anweisung

`USE FIXED AC MATRIX IN RERUNS` .

2.10.3 TR-Analyse

Für eine TR-Analyse ist *immer* die Angabe des Endzeitpunktes der Simulation erforderlich:

`STOP TIME =` *number* .

Weiterhin können folgende Größen gesteuert werden:

Anzahl der Integrationsschritte:

`MAXIMUM INTEGRATION PASSES =` *number* (20000)

Anfangszeitpunkt der TR-Analyse:

`START TIME =` *number* (0.0)

Auswahl des zu verwendenen Integrationsverfahrens:

```
INTEGRATION ROUTINE = { XPO | TRAP | RUK | IMPLICIT } .
```

Wird das implizite Integrationsverfahren gewählt, kann zusätzlich noch bestimmt werden:

```
USE { DIFFERENCED | SYMBOLIC } JACOBIAN .
```

Fehlt diese Anweisung, so wählt SCEPTRE automatisch den für das zu untersuchende Netzwerk besten Algorithmus aus.

SCEPTRE arbeitet mit variabler Integrationschrittweite, die nach jedem Integrationsschritt neu berechnet wird. Durch folgende Anweisungen können die Integrationsschrittweiten variiert werden:

Integrationsverfahren:	XPO	TRAP	RUK	IMPLICIT
MINIMUM STEP SIZE = *number*	10^{-5}	10^{-5}	10^{-5}	10^{-14}
MAXIMUM STEP SIZE = *number*	2×10^{-2}	2×10^{-2}	2×10^{-2}	2×10^{-2}
STARTING STEP SIZE = *number*	10^{-3}	10^{-3}	10^{-3}	10^{-8}
	$\times$ STOP TIME			

Die Integrationsgenauigkeit kann durch weitere Anweisungen beeinflusst werden. Die Bedeutung dieser Parameter wird in [3] beschrieben.

Integrationsverfahren:	XPO	TRAP	RUK	IMPLICIT
MINIMUM ABSOLUTE ERROR = *number*	10^{-4}	5×10^{-5}	5×10^{-5}	10^{-4}
MAXIMUM ABSOLUTE ERROR = *number*	5×10^{-3}	10^{-3}	5×10^{-3}	-
MINIMUM RELATIVE ERROR = *number*	2×10^{-4}	5×10^{-4}	5×10^{-5}	-
MAXIMUM RELATIVE ERROR = *number*	5×10^{-3}	10^{-2}	5×10^{-3}	-

2.10.4 DC- und TR-Analyse

Mit der Anweisung

```
RUN INITIAL CONDITIONS
```

wird zuerst eine DC-Analyse und anschließend eine TR-Analyse durchgeführt (`STOP TIME` nicht vergessen!). Die Ergebnisse der DC-Analyse werden als Anfangsbedingungen in die TR-Analyse eingesetzt[9]. Es können alle Steueranweisungen der DC- und TR-Analyse verwendet werden.

[9] Es werden immer die Ergebnisse der *letzten* DC-Analyse als Anfangsbedingungen eingesetzt. Die Toleranzanalysen werden unabhängig von der Reihenfolge der Steueranweisungen in der Reihenfolge SENSITIVITY, MONTE-CARLO, WORST-CASE und OPTIMIZATION durchgeführt.

2.10.5 DC- und AC-Analyse

Es gilt sinngemäß dasselbe wie bei der **DC- und TR-Analyse**. Es können alle Steueranweisungen der DC- und AC-Analyse benutzt werden.

2.10.6 AC- und TR-Analyse

Diese Kombination ist nicht möglich.

2.10.7 Ausgabesteuerung

Die unter `OUTPUTS` angeforderten Ergebnisse liegen zu jedem Integrationsschritt vor. Die Anzahl der tatsächlich auszudruckenden Ergebnisse läßt sich mit

`MAXIMUM PRINT POINTS =` *number* (1000)

steuern. Unterdrücken der Druckausgabe erreicht man, indem man *number* gleich Null setzt.

Zusätzlich kann man sich die Ergebnisse zu bestimmten Zeitintervallen ausdrucken lassen:

`PRINT INTERVAL =` *number* .

Mehrfachplotterzeichnungen werden unter `OUTPUTS` angefordert. Die Länge dieser Zeichnungen in Zeilen wird durch

`PLOT INTERVAL =` *number*

gesteuert. Sie ergibt sich aus dem Quotienten `STOP TIME` / `PLOT INTERVAL` und kann maximal 2000 Zeilen betragen. Fehlt diese Anweisung, wird für jede Ausgabegröße eine einfache Plotterzeichnung angefertigt.

2.10.8 Wiederholungsläufe

Bei Wiederholungsläufen (siehe Kapitel 2.13) werden die Anfangsbedingungen jeweils neu berechnet. Sie können aber auch explizit für jeden Lauf angegeben werden. Es besteht außerdem die Möglichkeit, als Anfangsbedingung die Ergebnisse des 1. Laufs oder vorherigen Laufs einzusetzen, und zwar mit der Anweisung

`IC FOR RERUNS =` { `MASTER` / `PRECEDING` } `RESULTS` .

2.10.9 Ablaufsteuerung

Man kann eine Simulation in Abhängigkeit vom Netzwerkverhalten abbrechen oder unterbrechen. Dies geschieht mit der Anweisung

{ TERMINATE / STOP } IF ({ *variable* / *constant* } *.ro.* { *variable* / *constant* }) .

Dabei steht *.ro.* für die Vergleichsoperatoren `.LE.`, `.LT.`, `.GT.`, `.GE.`, `.EQ.` und `.NE.`. Sie haben die gleiche Bedeutung wie in FORTRAN. Die Bedingungen können noch durch die logischen Operatoren `AND` und `OR` erweitert werden. Benutzt man `TERMINATE`, so wird der aktuelle Lauf abgebrochen, die evtl. noch anstehenden Wiederholungsläufe werden jedoch durchgeführt. Mit `STOP` wird der ganze Rechenlauf beendet. Beispiele:

```
TERMINATE IF (PWR .GT. 0.3)
STOP IF ((VRL .LE. E3) .OR. (IL9 .GT. 20.))
```

SCEPTRE schreibt während eines Rechenlaufs in regelmäßigen Zeitabständen interne Programmdaten in eine Datei. Endet ein Lauf abnorm (Systemzusammenbruch etc.), so braucht der Lauf nicht von Anfang an wiederholt zu werden, sondern kann mit den zuletzt gesicherten Programmdaten fortgesetzt werden (siehe Kapitel 2.14). Mit

`COMPUTER SAVE INTERVAL =` *number* (15)

kann das Zeitintervall geändert werden (Angaben in Minuten).

Eine weitere Vorsichtsmaßnahme bieten die Abfragen nach der bisher verbrauchten CPU-Zeit sowie der Verweilzeit des Programms im Rechner während der Ausführungsphase (elapsed time). Die maximale CPU-Zeit in Sekunden setzt man mit

`SOLUTION TIME LIMIT =` *number* .

Wenn diese Zeit verbraucht ist, druckt und plottet das Programm alle bis zu diesem Zeitpunkt errechneten Ergebnisse und endet ordnungsgemäß. Es wird nur die Rechenzeit der Ausführungsphase gezählt. Die maximale Verweilzeit in Minuten ändert man mit

`COMPUTER TIME LIMIT =` *number* (600) .

Die Verweilzeiten von Programmen hängen weitgehend von der Rechnerauslastung ab. Wird die angegebene Zeit während der Ausführungsphase überschritten, werden alle bis zu diesem Zeitpunkt vorhandenen Ergebnisse ausgegeben und das Programm ordnungsgemäß abgeschlossen.

Sollen die SCEPTRE-Anweisungen nur verarbeitet und nicht ausgeführt werden, z. B. zwecks Syntaxprüfung, erreicht man das mit

`EXECUTE SETUP PHASE ONLY` .

2.10.10 Testhilfen

Eine Übersicht aller Netzwerkknoten und dazugehörigen Zweige erhält man durch

`LIST NODE MAP` .

Die von SCEPTRE generierten FORTRAN-Programme werden ausgedruckt mit der Anweisung

`WRITE SIMUL8 DATA` .

Die Netzwerkmatrix, die die Topologie des Netzwerks beschreibt, erhält man mit

`PRINT B MATRIX` .

Bei einer AC-Analyse lassen sich mit den folgenden Anweisungen weitere Informationen ausdrucken:

`PRINT A MATRIX`

`PRINT EIGENVALUES`

`PRINT EIGENVECTORS` .

2.11 SENSITIVITY, MONTE CARLO, WORST CASE, OPTIMIZATION

Die Toleranzanalysen werden durch Anweisungen gleichen Namens aufgerufen.

`SENSITIVITY`

`MONTE CARLO`

`WORST CASE`

`OPTIMIZATION`

Jeder dieser Anweisungen folgt eine Parameterliste in der Form

`(`*abhängige Variablenliste/ unabhängige Variablenliste*`)` .

Die abhängigen Variablen definieren die Zielfunktion. Mehrere Variablen werden durch Kommata getrennt. Es müssen mindestens eine abhängige und eine unabhängige Variable vorhanden sein. Für die Monte-Carlo-, Worst-Case- und Optimization-Analysen müssen die unabhängigen Variablen mit Toleranzgrenzen angegeben werden. Tabellen 2.7 und 2.8 zeigen die Größen, die als abhängige und unabhängige Variablen auftreten können. Wird `P`*name* als abhängige Variable in der Sensitivity-, Optimization-, oder Worst-Case-Analyse benutzt, muss das vollständige Differential `GP`*name* ebenfalls angegeben werden. `P`*name* als unabhängige Variable kann nur als Faktor in einer Stromquelle 2. Ordnung auftreten, z. B. `J`*name1* `= P`*name* `* J`*name2*. Beispiel:

```
SENSITIVITY
  (IL3, PX / IX, EY)
  (VR1, IR1, VCX / P1, E1)
```

Tabelle 2.7: Abhängige Variablen

VC*name*	Spannung an einem Kondensator
IE*name*	Strom durch eine unabhängige Spannungsquelle
J*name*	abhängige Stromquelle 1. Ordnung
VJ*name*	Spannung an einer unabhängigen Stromquelle
IL*name*	Strom durch eine Induktivität
VR*name*	Spannung an einem Widerstand
IR*name*	Strom durch einen Widerstand
P*name*	definierter Parameter

Tabelle 2.8: Unabhängige Variablen

R*name*	Widerstand
E*name*	unabhängige Spannungsquelle
J*name*	unabhängige Stromquelle
P*name*	definierter Parameter

Es werden folgende partiellen Ableitungen berechnet:

$$\frac{\partial \mathtt{IL3}}{\partial \mathtt{IX}}, \frac{\partial \mathtt{IL3}}{\partial \mathtt{EY}}, \frac{\partial \mathtt{PX}}{\partial \mathtt{IX}}, \frac{\partial \mathtt{PX}}{\partial \mathtt{EY}}$$

$$\frac{\partial \mathtt{VR1}}{\partial \mathtt{P1}}, \frac{\partial \mathtt{VR1}}{\partial \mathtt{E1}}, \frac{\partial \mathtt{IR1}}{\partial \mathtt{P1}}, \frac{\partial \mathtt{IR1}}{\partial \mathtt{E1}}, \frac{\partial \mathtt{VCX}}{\partial \mathtt{P1}}, \frac{\partial \mathtt{VCX}}{\partial \mathtt{E1}}$$

Beispiel:

```
OPTIMIZATION
  (IR1, P1 / R1, P2, E1)
  (VC1 / I1, R2)
```

`IR1` wird optimiert bezogen auf `R1`, `P2` und `E1`.
`P1` wird optimiert bezogen auf `R1`, `P2` und `E1`.
`VC1` wird optimiert bezogen auf `I1` und `R2`.

2.12 Modelle

Netzwerkteile oder Ersatzschaltbilder, die häufig benutzt werden, können permanent oder temporär gespeichert werden und bei Bedarf in das zu untersuchende Netzwerk eingefügt werden. Dabei ist es möglich, einzelne Parameter im Modell zu ändern. Die Modellbeschreibung erfolgt unter der Überschrift

```
MODEL DESCRIPTION [ ( [ INITIAL, ] PRINT ) ]
```

und geht der Netzwerkbeschreibung voraus. Werden keine Modelle benutzt, kann dieser Paragraph entfallen. `INITIAL` wird nur dann benötigt, wenn das *erste Modell permanent* gespeichert werden soll. `PRINT` veranlasst, die Modellbeschreibung auszudrucken.

Die Befehle `PRINT` und `INITIAL` sind nur für permanent gespeicherte Modelle erlaubt.

2.12.1 Modellbeschreibung

Folgende Anweisungen sind für eine Modellbeschreibung erlaubt:

```
MODEL modelname [ ( { PERM } ) ] ( node1 – node2 – … )
                   { TEMP }
```

PERM bedeutet, dass das Modell permanent gespeichert werden soll. Mit TEMP wird es am Ende der Simulation wieder gelöscht. *modelname* kann maximal 18 Zeichen lang sein. Ein Modell darf bis zu 25 Anschlußpunkte (*node1 – node2 – …*) haben. Mit

```
MODEL modelname (DELETE)
```

wird ein permanent gespeichertes Modell gelöscht.

Nach der MODEL Anweisung können bis zu 11 Kommentarzeilen folgen. Danach sind Anweisungen zu folgenden Überschriften möglich:

- ELEMENTS
- DEFINED PARAMETERS
- OUTPUTS
- FUNCTIONS
- INITIAL CONDITIONS.

Für die dazugehörigen Anweisungen gelten die gleichen Regeln wie unter CIRCUIT DESCRIPTION. Beispiel:

```
MODEL DESCRIPTION (INITIAL,PERM)
MODEL 2N1734B (PERM) (B-E-C)
  ELEMENTS
    CE, B-E = Q1 (5., 80., TABLE 1 (VCE))
    CC, B-C = Q1 (10., 200., TABLE 2 (VCC))
    J1, B-E = DIODE TABLE 1
    JA, E-B = 0.1 * J2
    J2, B-C = DIODE TABLE 2
    JB, C-B = 0.98 * J1
  OUTPUTS
    VCE, VCC, J1, PLOT
  FUNCTIONS
    DIODE TABLE 1 = 0,0, .3,0, .65,.5, ....
    DIODE TABLE 2 = 0,0, .58,0, .62,.4, ....
    EQUATION 1 (A,B,C) = (A + B * C)
```

Ob es sinnvoll ist, die OUTPUTS-Anweisung in die Modellbeschreibung einzufügen, muss man von Fall zu Fall entscheiden.

2.12.2 Modellaufruf

Ein Modell wird unter `CIRCUIT DESCRIPTION/ELEMENTS` mit folgender Anweisung in das Netzwerk eingefügt:

name, node1 – node2 – … `= MODEL` *modelname* [`(` { `PERM` | `TEMP` } `)`] …

…[`,` `CHANGE` { *element* | *def.-par.* } `=` *value* [`,` { … | … } `=` *value*]] …

…[`,` `SUPPRESS` { *var1, var2, …* | `ALL` }] [`PRINT`] `)` .

`PERM` oder `TEMP` gibt an, ob es sich um ein permanent oder temporär gespeichertes Modell handelt. *modelname* ist der Name des Modells, unter dem es gespeichert oder definiert wurde.

Alle internen Modellnamen (Elemente, Knotenpunkte, usw.) werden um den Bezugsnamen *name* erweitert (siehe folgendes Beispiel). Damit ist jeder interne Modellname eindeutig gekennzeichnet, auch wenn das gleiche Modell mehrere Male eingefügt wird. SCEPTRE behandelt dann dieses Modell so, als ob es direkt unter `CIRCUIT DESCRIPTION` definiert worden wäre. Da unter `CIRCUIT DESCRIPTION` die Namenslänge begrenzt ist, empfiehlt es sich, die im Modell internen, frei wählbaren Namen sowie den Bezugsnamen möglichst kurz zu wählen entsprechend der Angaben in Tabelle 2.2. Beispiel: Aufruf des o. a. Modells 2N1734B:

```
T1, 13-4-GND = MODEL 2N1734B (PERM)
```

Will man sich auf eine Größe in dem eingefügten Modell T1 beziehen, muss diese Größe um den Bezugsnamen `T1` erweitert werden, z.B. `CET1, CCT1, J1T1, …, VCET1, ICET1,` usw..

Mit `CHANGE` können die Werte von Netzwerkelementen und definierten Parametern in dem eingefügten Modell geändert werden. Die folgenden Beispiele beziehen sich wieder auf das Modell 2N1734B.

```
T1, 13-4-GND = MODEL 2N1734B (PERM, CHANGE CC=50)
T1, 13-4-GND = MODEL 2N1734B (PERM, CHANGE CC=TABLE 7(VCCT1),
               JA=0.2*J2T1, JB=DIODE TABLE 4)
```

Die Werte für die Kapazität `CC` ergeben sich aus einer neuen Tabelle (`TABLE 7`), die Stromquelle `JB` wird mittels einer Diodenkennlinie berechnet (`DIODE TABLE 4`). Beide Tabellen müssen unter `CIRCUIT DESCRIPTION/FUNCTIONS` definiert sein. Die gesteuerte Stromquelle `JA` wird mit einer anderen mathematischen Anweisung berechnet.

```
T1, 13-4-GND = MODEL 2N1734B (PERM,CHANGE CC=EQUATION 5 (VCCT1,VCX))
```

Die Kapazitätswerte für `CC` werden durch eine neue Formelfunktion (`EQUATION 5`) berechnet, die unter `CIRCUIT DESCRIPTION/FUNCTIONS` definiert sein muss. Dabei werden als Parameter die Spannungen an den Kondensatoren `CC` und `CX` übergeben. Der Kondensator `CX` gehört nicht zu dem Modell 2N1734B.

```
T1, 13-4-GND = MODEL 2N1734B (PERM, CHANGE TABLE 1 = TABLE 7)
```

Die Tabelle 7 ersetzt die Tabelle 1 in dem Modell und muss unter `CIRCUIT DESCRIPTION/FUNCTIONS` definiert werden. Die Anzahl der Punktepaare in beiden Tabellen braucht nicht übereinzustimmen[10].

```
T1, 13-4-GND = MODEL 2N1734B (PERM, CHANGE Q1=Q2)
```

Es wird die Funktion `EQUATION 1` durch eine neue Funktion `EQUATION 2` ersetzt, die unter `CIRCUIT DESCRIPTION/FUNCTIONS` definiert sein muss. Die neue Funktion muss die gleichen Parameter haben[11]. Die Ausgabe von Ergebnissen, die man unter `MODEL DESCRIPTION/OUTPUTS` anfordern kann, lässt sich ganz oder teilweise mit dem Befehl `SUPPRESS` unterdrücken.

```
T1, 13-4-GND = MODEL 2N1734B (PERM, SUPPRESS J1)
T1, 13-4-GND = MODEL 2N1734B (PERM, SUPPRESS ALL)
```

Mit dem Befehl `PRINT` wird die Modellbeschreibung aufgelistet.

```
T1,13-4-GND=MODEL 2N1734B(PERM,CHANGE CC=50,CE=30,SUPPRESS J1,VCE,PRINT)
```

Die beim Modellaufruf möglichen Befehle können miteinander kombiniert werden. Anfangsbedingungen setzt man vorteilhafter unter `CIRCUIT DESCRIPTION/INITIAL CONDITIONS`.

```
VCCT1 = 0.53
```

2.13 RERUN DESCRIPTION

SCEPTRE bietet die Möglichkeit, einen bereits durchgeführten Lauf mit veränderten Parametern zu wiederholen. Dabei brauchen nur noch die sich ändernden Werte angegeben zu werden. Diese Änderungen werden unter der Überschrift

`RERUN DESCRIPTION` [`(` *n* `)`]

zusammengefaßt. Die Zahl *n* gibt die Anzahl der modifizierten Rechenläufe an. Der Klammerausdruck entfällt bei nur einem Wiederholungslauf. Im Gegensatz zur `CIRCUIT DESCRIPTION` dürfen bei der `RERUN DESCRIPTION` nur die Werte *einer* Variablen pro Eingabezeile geändert werden.

Die konstanten Variablen unter `ELEMENTS` ändert man mit

elementname `=` *number1, number2, …* .

Im 1. Wiederholungslauf wird *elementname* der Wert *number1*, im 2. Lauf der Wert *number2* usw. zugewiesen.

Entsprechend werden den konstanten Variablen unter `DEFINED PARAMETERS` neue Werte zugeordnet:

def. par. name `=` *number1, number2, …*

Beispiel:

[10] Achtung: es ist *nicht möglich*, eine Diodenkennlinie direkt in eine andere Diodenkennlinie abzuändern (`CHANGE DIODE TABLE 1 = DIODE TABLE 9`).

[11] Achtung: es ist *nicht möglich*, in einem Modell eine Funktion in eine Tabelle oder umgekehrt abzuändern (`CHANGE EQUATION 1 = TABLE 4`). Es können dagegen Netzwerkelemente und definierte Parameter durch Konstanten, Tabellen oder Funktionen ersetzt werden.

```
RERUN DESCRIPTION (3)
  ELEMENTS
    RLOAD = 25, 200, 20E3
  DEFINED PARAMETERS
    PSI = 13.4, 13.2, 13.1
```

Ebenso können Toleranzgrenzen neu definiert werden:

{ *def. par. name* / *element name* } = *number1* ({ *number2, number3* / *number4* }) , …

Beispiel:

```
R3 = 4.6, 9.(8.,10.5)
```

Im 1. Lauf wird dem Widerstand `R3` der Wert `4.6` ohne Toleranzgrenzen zugewiesen, im 2. Lauf der Nominalwert `9.` mit der Unter- und Obergrenze von `8.` bzw. `10.5`.

Die Anfangsbedingungen unter `INITIAL CONDITIONS` können geändert werden mit

{ `VC` / `IL` / `VJ` } *name* = *number1, number2, …* .

Unter `FUNCTIONS` können die Tabellenwerte geändert werden. Die neue Tabelle muss die gleiche Länge besitzen wie die ursprüngliche Tabelle. Pro `RERUN DESCRIPTION`-Anweisung können die unabhängigen (x-Werte) einer Tabelle nur einmal überschrieben werden. Die dazugehörenden Tabellenwerte (y-Werte) können beliebig oft geändert werden. Die allgemeine Anweisung lautet:

`TABLE` *name* = *X1, Y11, Y12,…Y1N, X2, Y21, Y22,…Y2N,…* .

Der Index *N* entspricht der Anzahl der gewünschten Wiederholungsläufe.

Beispiel:

```
CIRCUIT DESCRIPTION
  .....
  FUNCTIONS
    TABLE 7 = 0,0, 1,10, 2,27, 4,47
  .....
RERUN DESCRIPTION (2)
  FUNCTIONS
    TABLE 7 = 0,0,0, 1,3,5, 3,4,8, 5,7,9
```

Tabelle 7 wird dann folgendermaßen belegt:

Originallauf		1. Wiederholung		2. Wiederholung	
X	Y	X	Y	X	Y
0	0	0	0	0	0
1	10	1	3	1	5
2	27	3	4	3	8
4	47	5	7	5	9

Man erkennt, dass die x-Werte der Tabelle im 2. Wiederholungslauf die gleichen Werte aufweisen wie im ersten Lauf. Will man die x-Werte bei der 2. Wiederholung ebenfalls ändern, muss man eine zweite `RERUN DESCRIPTION` Überschrift einfügen.

Beispiel:

```
RERUN DESCRIPTION
  FUNCTIONS
    TABLE 7 = 0,0, 1,3, 3,4, 5,7
RERUN DESCRIPTION
  FUNCTIONS
    TABLE 7 = 0,0, 2,4, 3,5, 4,9
```

Unter `RUN CONTROLS` können alle Anweisungen geändert werden mit Ausnahme von `STOP` und `TERMINATE`. Bezüglich der Wahl der Analysearten gibt es einige Einschränkungen:

- Die Topologie eines Netzwerkes kann nicht geändert werden.
- Die Angaben unter `OUTPUTS` können nicht geändert werden.
- Konstanten müssen durch Konstanten, Tabellen durch Tabellen und `EQUATION`-Anweisungen durch `EQUATION`-Anweisungen ersetzt werden.
- `EXPRESSION`- und `EQUATION`-Anweisungen können nur dann geändert werden, wenn die in ihnen enthaltenen Variablen unter `DEFINED PARAMETERS` definiert werden. Man ändert dann diese Parameter.

  Beispiel:

  ```
  CIRCUIT DESCRIPTION
    ELEMENTS
      JE, 1-8 = DIODE EQUATION (PX1,PX2)
      .....
    DEFINED PARAMETERS
      PX1 = ...., PX2 = .....
      .....
  RERUN DESCRIPTION
    DEFINED PARAMETERS
      PX1 = ....
      PX2 = ....
  ```

- Die Änderungen von Netzwerkelementen erfordern u. U. eine Neuberechnung der Anfangsbedingungen.
- Im allgemeinen kann die Analyseart nicht geändert (gewechselt) werden.

- Es werden keine neuen Anfangsbedingungen berechnet, auch wenn sie im ursprünglichen Lauf durchgeführt wurden. Um dies auch für die Wiederholungsläufe zu erreichen, muss unter `RUN CONTROLS` die Anweisung `RUN INITIAL CONDITIONS` erneut angegeben werden. Es werden dann bei *jedem* Wiederholungslauf die Anfangsbedingungen berechnet.
- Folgende Steueranweisungen sind in Wiederholungsläufen nur gültig, wenn die Anfangsbedingungen im ursprünglichen Lauf berechnet wurden:
  - `RUN SENSITIVITY`
  - `RUN MONTE CARLO`
  - `RUN WORST CASE`
  - `RUN OPTIMIZATION`
  - `RUN INITIAL CONDITIONS`
  - `RUN INITIAL CONDITIONS ONLY`.
- Die folgenden Anweisungen sind in Wiederholungsläufen nicht erlaubt:
  - `NO ELEMENT SORT`
  - `PUNCH PROGRAM`
  - `WRITE SIMUL8 DATA`
  - `VECTOR EQUATIONS`
  - `WRITE DEBUG`
  - `PRINT B MATRIX`
  - `LIST NODE MAP`
  - `IC FOR RERUNS`
  - `USE DIFFERENCED JACOBIAN`
  - `USE SYMBOLIC JACOBIAN`
  - `EXECUTE SETUP PHASE ONLY`
  - `USE FIXED AC MATRIX IN RERUNS`
  - `IMPULSE RESPONSE BUFFER`
  - `INPUT FUNCTION BUFFER`.
- Folgende Anweisungen werden vom ursprünglichen Lauf nicht mit in den Wiederholungslauf übernommen, sie müssen bei Bedarf neu angegeben werden:
  - `RUN AC`
  - `RUN SENSITIVITY`
  - `RUN MONTE CARLO`
  - `RUN WORST CASE`
  - `RUN OPTIMIZATION`
  - `LIST MONTE CARLO DETAILS`

  - `LIST OPTIMIZATION DETAILS`
  - `PUNCH OPTIMIZATION RESULTS`
  - `OPTIMIZATION RANDOM STEPS`
  - `PRINT A MATRIX`
  - `PRINT EIGENVALUES`
  - `MINIMUM FUNCTION ESTIMATE`
  - `INITIAL RANDOM NUMBER`.

- Folgende Anweisungen widersprechen sich und führen zu einem Programmabbruch:

```
CIRCUIT DESCRIPTION
  .......
  IC FOR RERUNS = MASTER RESULTS
  oder
  IC FOR RERUNS = PRECEDING RESULTS
RERUN DESCRIPTION
  ......
  RUN INITIAL CONDITIONS [ ONLY ]
  oder
  RUN IC VIA IMPLICIT
```

Für Änderungen in Modellen gelten die gleichen Regeln. Es ist nur darauf zu achten, dass die Elementnamen um den Bezugsnamen erweitert werden.

Beispiel:

```
MODEL DESCRIPTION
  MODEL ABCDEFG (PERM) (1-7-8-9)
    ELEMENTS
      ....
      R1, 1-2 = 1
    DEFINED PARAMETERS
      P1 = 2.7
    FUNCTIONS
      TABLE 1 = ...
CIRCUIT DESCRIPTION
  ELEMENTS
    ...
    T8, 4-9-1-2 = MODEL ABCDEFG
    ...
RERUN DESCRIPTION
  ELEMENTS
    R1T8 = 0.98
  DEFINED PARAMETERS
    P1T8 = 3.1
  FUNCTIONS
    TABLE 1 T8 = ...
```

2.14 CONTINUE

Die CONTINUE-Anweisung erlaubt dem Benutzer, einen ordnungsgemäß abgeschlossenen Rechenlauf fortzusetzen[12]. Sie kann nur auf TR-Analysen angewandt werden. Bei Wiederholungsläufen wird der letzte durchgeführte Lauf fortgesetzt. Eine Änderung im zu untersuchenden Netzwerk ist nicht möglich. Es können nur Anweisungen unter `RUN CONTROLS` geändert werden, und zwar:

- `STOP TIME`
- `COMPUTER TIME LIMIT`
- `MAXIMUM INTEGRATION PASSES`
- `INTEGRATION ROUTINE`
- `USE …JACOBIAN`
- `MINIMUM/MAXIMUM STEP SIZE`
- `STARTING STEP SIZE`
- `MINIMUM/MAXIMUM ABSOLUTE ERROR`
- `MINIMUM/MAXIMUM RELATIVE ERROR`
- `MAXIMUM PRINT POINTS`
- `PRINT INTERVAL`
- `PLOT INTERVAL`

Beispiel:

```
CONTINUE
  RUN CONTROLS
    STOP TIME = 200
END
```

2.15 RE-OUTPUT

Eine weitere Möglichkeit, Rechenzeit zu sparen, bietet die `RE-OUTPUT`-Anweisung. Sie erlaubt es, zusätzliche Ausdruckkopien zu einem vorher abgeschlossenen Rechenlauf zu erzeugen, ohne diesen Lauf noch einmal zu wiederholen. Dabei kann die Ausgabe der Ergebnisse in Grenzen gesteuert werden. Unter `OUTPUTS` wird eine Liste der Ausgabevariablen angegeben, die ausgedruckt bzw. geplottet werden sollen. Diese Liste kann nur Ausgabevariablen enthalten, die im ursprünglichen Lauf unter `OUTPUTS` angefordert wurden[13]. Zusätzlich kann unter `RUN CONTROLS` die Anzahl der Ausgabepunkte mit `MAXIMUM PRINT POINTS` gesteuert werden.

Bei der AC-Analyse ist außerdem darauf zu achten, dass Plotterzeichnungen nur in den Einheiten ausgegeben werden können[14], die im ursprünglichen Lauf angefordert wurden.
Beispiel:

[12] siehe auch Kapitel 2.10.9: Ablaufsteuerung
[13] unabhängig von der Art der Ausgabe
[14] `RADIANS, DEGREES, COMPLEX`

```
RE-OUTPUT
  OUTPUTS
    VR1, VR2
    VL1, VCC, PLOT
  RUN CONTROLS
    MAXIMUM PRINT POINTS = 1500
END
```

2.16 END

Die Anweisung `END` markiert das Ende der SCEPTRE Eingabe und muss als letzte Anweisung stehen.

3 Allgemeine Hinweise

Dieses Kapitel enthält wichtige Hinweise zur Benutzung von SCEPTRE. Außerdem wird auf mögliche Fehlerquellen und deren Ursache eingegangen.

3.1 Fehlerdiagnose

SCEPTRE verfügt über eine umfangreiche Fehlerdiagnose. Die Fehlermeldungen werden in 3 Klassen eingeteilt:

warning only: es handelt sich dabei im allgemeinen um unkritische Fehler, z. B. Syntaxfehler, die SCEPTRE berichtigen kann; die Simulation wird ausgeführt;

simulation deleted: Fehler dieser Kategorie verhindern eine anschließende Simulation; die Eingabedaten werden jedoch weiter geprüft;

execution terminated: bei schwerwiegenden Fehlern wird die Fehlerdiagnose abgebrochen.

Neben Syntaxfehlern und elementaren Verstößen gegen die Eingabestruktur werden auch Fehler aufgedeckt, die sich aus der Gesamtheit der Schaltung ergeben, z. B. „computational delay“, unzulässige Netzwerktopologien, etc.. Im allgemeinen sind die Fehlernachrichten so gezielt, dass man unschwer die Ursache entdecken kann. Es gibt aber auch Fälle, bei denen eine unrichtige Schreibweise (z. B. die Null statt des Buchstabens „O“) eine Flut von verwirrenden Fehlermeldungen auslösen kann.

Neben diesen Eingabefehlern gibt es auch logische Fehler, die man erst anhand der Ergebnisse erkennt. Hier kann SCEPTRE keine Hilfe bieten[1].

3.2 Einige Grundregeln

- Physikalische Einheiten sollten so gewählt werden, dass sich die maximalen und minimalen Werte um nicht mehr als 5–6 Zehnerpotenzen unterscheiden und sich in der Nähe von 1 bewegen. Weit auseinander liegende Werte verschlechtern oder verhindern sogar die Konvergenz der iterativen Verfahren.
- Werden Modelle benutzt (z. B. aus einer permanenten Bibliothek), muss unbedingt kontrolliert werden, ob das Modell und die Schaltung, in die das Modell eingefügt werden soll, die gleichen Maßeinheiten verwenden.
- Bei größeren bzw. komplexen Schaltungen empfiehlt es sich, die Schaltung schrittweise in Betrieb zu nehmen und anhand von Einzelsimulationen die Richtigkeit der Modelle und Ersatzschaltungen zu überprüfen.

[1] Es stehen jedoch eine Reihe von Anweisungen zur Verfügung, die in manchen Situationen hilfreich sein können (siehe Kapitel 2.10.10: Testhilfen).

- Nichtlinearitäten sollten möglichst durch gesteuerte Quellen beschrieben werden statt passive Elemente (R,C,L) zu benutzen.
- Bei spannungsabhängigen Nichtlinearitäten sollte die Spannung an einem Kondensator, bei stromabhängigen Nichtlinearitäten der Strom durch eine Induktivität zur Steuerung verwendet werden, da Kondensatorspannungen und Induktivitätsströme Zustandsvariablen sind und mit jedem Integrationsschritt neu berechnet werden. Damit beugt man einem „computational delay“ vor (siehe auch Kapitel 3.7: „Computational Delay“).

3.3 Zeitabhängige Induktivitäten und Kapazitäten

Der verwendete Lösungsansatz liefert falsche Resultate, wenn bei der TR-Analyse Induktivitäten bzw. Kapazitäten von der Zeit abhängen (siehe auch [1, Kapitel 1.7]). Gilt für eine Kapazität $C = f(t)$, muss eine Stromquelle $J = V_C \frac{\mathrm{d}\,C}{\mathrm{d}\,t}$ dem Kondensator parallel geschaltet werden. Für eine zeitabhängige Induktivität $L = f(t)$ korrigiert eine in Reihe geschaltete Spannungsquelle $E = I_L \frac{\mathrm{d}\,L}{\mathrm{d}\,t}$ den Fehler. Zeitabhängige Widerstände, stromabhängige Induktivitäten und spannungsabhängige Kapazitäten werden richtig berechnet.

3.4 Tabellen

Werden Tabellen zur Beschreibung von Nichtlinearitäten benutzt, empfiehlt es sich, die Kurven möglichst zu glätten bzw. zusätzliche Stützstellen einzufügen, um Knicke zu vermeiden. Besonders die ersten und letzten Wertepaare sind kritisch, da SCEPTRE außerhalb liegende Werte aus diesen Wertepaaren extrapoliert. Beschreibt man passive Elemente (R,C,L) mit Tabellen, ist darauf zu achten, daß diese für *keinen* unabhängigen Wert Null werden dürfen. Grundsätzliche sollte man geschlossene Funktionen Tabellen vorziehen.

Werden Tabellen innerhalb einer `EXPRESSION-` oder `EQUATION-`Anweisung verwendet, muss man eine andere Syntax benutzen:

```
..... = X4711 ( ..... XTABLE(Tname,variable) ..... )
```

Für `name` wird der Tabellenname eingesetzt, die unabhängige Variable *muss* als zweiter Parameter angegeben werden, z. B. `TIME`.

3.5 Arithmetische Anweisungen

Verwendet man arithmetische Anweisungen zur Beschreibung abhängiger Elemente oder Parameter, sollte man den Gültigkeitsbereich der Formeln überprüfen. Die Standardfunktionen `MIN, MAX, ABS` etc. in FORTRAN sind z. B. gut geeignet, Divisionen durch Null oder Überläufe zu verhindern. Beispiele:

```
CC, B-5 = X1 (PX/(0.8-MIN(0,VCC))**0.33333)
JD, C-B = X2 (PS*EXP(MIN(88,PQ*VJD))-1)
```

3.6 Topologische Einschränkungen

Enthält ein Netzwerk Maschen, bestehend aus Spannungsquellen und Kapazitäten, bzw. Knotenpunkte, die nur über Stromquellen und Induktivitäten erreichbar sind, muss bei variablen Quellen die erste zeitliche Ableitung vorgegeben werden.

3.6.1 DC-, AC- und TR-Analyse

Bei keiner der drei Analysearten dürfen in dem Netzwerk Maschen vorhanden sein, die ausschließlich aus Spannungsquellen bestehen (*voltage source loop*). Ebenso sind Knotenpunkte verboten, die nur über Stromquellen zu erreichen sind (*all-current source cut set*).

3.6.2 DC-Analyse

Sollen Anfangsbedingungen berechnet werden, dürfen in dem Netzwerk keine Maschen auftreten, die nur aus Spannungsquellen und Induktivitäten bestehen. Ebensowenig sind Knotenpunkte zugelassen, die nur über Kapazitäten und Stromquellen zu erreichen sind. Dem Problem im letzten Fall kann man dadurch begegnen, dass man den fraglichen Knotenpunkt über einen Widerstand, der das Verhalten der Schaltung nicht beeinflußt, z. B. mit Masse verbindet.

Man kann diese Einschränkung umgehen, indem man die Anfangsbedingungen nicht über eine Gleichstromanalyse sondern über eine Transientanalyse berechnet. Die notwendige Syntax lautet dann:

```
RUN INITIAL CONDITIONS [ ONLY ]
RUN IC VIA IMPLICIT
```

Man kann dieses Verfahren auch anwenden, falls die Gleichstromanalyse nicht konvergiert (siehe Kapitel 3.17).

3.7 „Computational Delay“

Eine der überragenden Fähigkeiten von SCEPTRE besteht in der Modellierung von Nichtlinearitäten. Dies wird u. a. dadurch ermöglicht, dass der Benutzer jedes Netzwerkelement in Abhängigkeit von anderen Größen beschreiben kann. SCEPTRE berechnet zu jedem Integrationsschritt die zeitlichen Ableitungen der unabhängigen Variablen (Zustandsvariablen) und daraus mit den Netzwerkgleichungen die anderen Netzwerkvariablen. Voraussetzung ist dafür allerdings, dass die unabhängigen Variablen zu Beginn eines Integrationsschrittes bekannt sind. Andernfalls verwendet SCEPTRE den Wert der unabhängigen Variablen vom vorherigen Integrationsschritt. Es wird jedoch eine Warnung

```
THE TERM '......' WILL CAUSE A COMPUTATIONAL DELAY
```

ausgegeben. Die Größe des Fehlers lässt sich schwer abschätzen und ist von dem Grad der Nichtlinearität der Funktion und der Integrationsschrittweite abhängig. Dieser Integrationsfehler tritt nicht bei Abhängigkeiten von der Zeit, Spannungen an Kondensatoren und Strömen durch Induktivitäten auf, da diese Größen immer unabhängige Variablen sind. Auch Widerstandsspannungen und -ströme können unabhängig sein, vorausgesetzt, dass sie mit der richtigen Syntax[2] definiert werden.

[2] siehe Kapitel 2.5.3, 2.5.4 sowie 3.9

Dieses Problem tritt z. B. bei Diodenstromquellen auf, die spannungsgesteuert sind. Durch Parallelschaltung einer Kapazität — die ja in Wirklichkeit auch vorhanden ist — bildet die Kondensatorspannung die unabhängige Variable.

Falls diese Methode versagt, kann man die von SCEPTRE generierten FORTRAN Programme untersuchen. Mit der Anweisung `WRITE SIMUL8 DATA` unter `RUN CONTROLS` druckt SCEPTRE das Unterprogramm `SIMUL8` aus. Dem Listing kann man die Reihenfolge entnehmen, in der die abhängigen Variablen berechnet werden. Unter Umständen ist es möglich, die in Frage kommende Größe durch Netzwerkvariablen zu beschreiben, die zu diesem Integrationszeitpunkt bereits bekannt sind (siehe auch Kapitel 3.13). Lautet die Beschreibung unter `ELEMENTS` z. B.

```
L2, ... = X1 (P1*IR2)
```

und erhält man die Warnung

```
THE TERM 'IR2' WILL CAUSE A COMPUTATIONAL DELAY
```

kann man diese Warnung vermeiden, in dem man z. B. `L2` als Funktion von `VR2` definiert, vorausgesetzt, `VR2` ist zu diesem Zeitpunkt bereits berechnet worden:

```
L2, ... = X1 (P1*VR2/R2)
```

3.8 Wahl des Integrationsverfahrens

SCEPTRE bietet wahlweise drei explizite Integrationsverfahren an: das Exponentialverfahren (`XPO`), das Runge-Kutta-Verfahren 4. Ordnung (`RUK`) und ein modifiziertes Trapezverfahren (`TRAP`), sowie ein implizites (`IMPLICIT`). Das Exponentialverfahren, eine Eigenentwicklung von IBM, stellt einen Kompromiss zwischen Schnelligkeit und Genauigkeit dar. Es wird deshalb automatisch benutzt, falls der Benutzer kein Verfahren angibt. Das implizite Verfahren wird vorteilhaft immer dann angewandt, wenn in einem Netzwerk gleichzeitig sehr große und sehr kleine Zeitkonstanten auftreten.

Jedes Integrationsverfahren hat für die Schrittweitenbegrenzung und die Fehlerabschätzung einen Satz eigener Parameter[3]. Nach jedem Integrationsschritt wird der Fehler abgeschätzt und daraus die neue Schrittweite berechnet. Wird dabei die unterste Schranke unterschritten, bricht das Programm ab mit der Fehlermeldung

```
SMALLER MINIMUM STEP SIZE REQUIRED
```

und dem Index der verursachenden Zustandsvariablen[4]. Diese Fehler treten auf, wenn die Zeitkonstanten des Netzwerkes sehr viel kleiner sind als die gesamte Simulationszeit (`STOP TIME`). Man kann den Fehler dadurch beheben, dass man

- die Simulationszeit verkleinert,
- die Elemente, die zur Bildung der Zeitkonstanten beitragen, überprüft und die Zeitkonstanten evtl. vergrößert,

[3] siehe Kapitel 2.10.3: TR-Analyse
[4] SCEPTRE listet stets alle Zustandsvariablen zusammen mit dem zugehörigen Index auf.

- die minimale Schrittweite verkleinert (`MINIMUM STEP SIZE`), voraussichtlich wird man die maximale Anzahl der Integrationsschritte vergrößern müssen (`MAXIMUM INTEGRATION PASSES`).

Treten in einer Schaltung Impulse mit sehr geringer Impulsbreite auf, so muss die Schrittweite soweit verringert werden, dass SCEPTRE diesen Impuls erkennen kann.

Es empfiehlt sich — vor allem bei rechenintensiven Simulationen — vorher Testläufe mit jeweils einem der vier Integrationsverfahren durchzuführen und die Ergebnisse sowie die Rechenzeit kritisch miteinander zu vergleichen.

Es gibt Simulationsprobleme, bei denen das eine oder andere Integrationsverfahren versagt oder unverhältnismäßig viel Rechenzeit benötigt. Auf die Problematik der numerischen Integrationsverfahren wird in [3] und besonders in [6, Kapitel 3.5] eingegangen.

3.9 Definition abhängiger Quellen

Es ist wichtig, dass abhängige Quellen richtig in SCEPTRE definiert werden, damit sie verzögerungsfrei bei einer Transientanalyse zur Verfügung stehen[5]. Es gibt vier Arten von abhängigen Quellen:

1. widerstandsabhängige Spannungsquellen:
   `Exx, node1 - node2 = constant * VRyy`
2. widerstandsabhängige Stromquellen:
   `Jxx, node1 - node2 = constant * IRyy`
3. Dioden (Stromquellen 1. Ordnung):
   `Jxx, node1 - node2 = DIODE EQUATION (x1,x2)`
   `Jxx, node1 - node2 = DIODE TABLE xy`
4. diodenabhängige Stromquellen (2. Ordnung):
   `Jxx, node1 - node2 = variable * Jyy`
   wobei `Jyy` eine Stromquelle 1. Ordnung sein muss.

Diese Quellen müssen exakt so definiert werden, andernfalls erkennt SCEPTRE diese Quellen nicht richtig. So kann z. B. die Schreibweise

```
Exx, node1 - node2 = X1 (constant * VRyy)
```

zu einem "computational delay" führen, da SCEPTRE die Spannungsquelle `Exx` nicht als widerstandsabhängig erkennt. Die richtige Definition von abhängigen Quellen führt im allgemeinen zu kürzeren Rechenzeiten bei der DC- und TR-Analyse. *Nicht richtig definierte Quellen werden bei der AC-Analyse wie Gleichstromquellen behandelt und auf Null gesetzt.* Vielfach lässt sich eine anders definierte abhängige Quelle in eine Quelle der o. a. Form umwandeln. Z. B. statt

```
E1, N1 - N2 = X1 (3. * VC1)
C1, NA - NB = ....
```

kann man einen großen Widerstand parallel zu C1 schalten und dann schreiben

[5] siehe Kapitel 3.7: „Computational Delay“

```
E1, N1 - N2 = 3. * VR1
C1, NA - NB = .....
R1, NA - NB = grosser Wert
```

Ähnlich kann man eine stromabhängige Spannungsquelle

```
E1, N1 - N2 = X1 (3. * IR1)
R1, NA - NB = 2.
```

in die richtige Form bringen unter Einbeziehung des Widerstandes R1.

```
E1, N1 - N2 = 1.5 * VR1
R1, NA - NB = 2.
```

3.10 Strom- und Spannungsmessungen

Den Strom durch eine Komponente erhält man, indem man dem Komponentennamen den Buchstaben `I` voranstellt. Sinngemäß erhält man die Spannung an einer Komponente durch Voranstellen des Buchstabens `V`[6]. Will man Spannungen zwischen weit auseinanderliegenden Knotenpunkten messen, kann man mit Hilfe einer Nullstromquelle zwischen diesen Punkten die Spannung messen, ohne die Schaltung zu beeinflussen.

```
Jxx, NA - NZ = 0
VJxx, PLOT
```

Entsprechend lassen sich auch Ströme in Zweigen mit einer Nullspannungsquelle messen, ohne dass ein Shuntwiderstand eingefügt werden muss.

```
Exx, NM - NN = 0
IExx, PLOT
```

3.11 Verwendung eigener FORTRAN-Unterprogramme

Das Einfügen eigener FORTRAN Unterprogramme gibt dem Benutzer Gelegenheit, spezielle Berechnungen durchzuführen, die nicht direkt in SCEPTRE möglich sind. Es können Werte zwischen SCEPTRE und dem Unterprogramm in beiden Richtungen ausgetauscht werden. FORTRAN-Unterprogramme ruft man mit der SCEPTRE-Anweisung

Netzwerkvariable `=` `F` *name (Parameterliste)*

auf. Im allgemeinen wird *Netzwerkvariablen* ein Wert zugewiesen.

Der Unterprogrammname `F`*name* darf maximal 6 Zeichen lang sein. Das entsprechende FORTRAN Unterprogramm muss mit der folgendenAnweisung beginnen:

`DOUBLE PRECISION FUNCTION F` *name (Parameterliste)* .

[6] vgl. Tabelle 2.1

Alle in der Parameterliste vorkommenden Variablen müssen doppelt genau definiert werden[7]. Zusätzlich können Unterprogramme über den COMMON-Block `/CNTRLS/`[8] interne SCEPTRE-Variablen abfragen. **Achtung:** die SCEPTRE-Variablen in der Parameterliste dürfen in dem Unterprogramm nicht verändert werden.

Möchte man bei jedem Integrationsschritt FORTRAN-Unterprogramme aufrufen ohne Daten auszutauschen, um z. B. Daten in eine externe Datei zu schreiben oder eine eigene Graphik zu erzeugen, muss man auch die obige Schreibweise verwenden und eine Hilfsvariable benutzen, z. B.

```
DEFINED PARAMETERS
  PDUMM = Fname (Parameterliste)
```

SCEPTRE prüft beim Generieren der FORTRAN-Anweisungen, wo die Variable `PDUMM` benötigt wird. Gibt es keine Verwendung für die Variable, wird die Anweisung zum Berechnen von `PDUMM` nicht ausgeführt. *Damit das Unterprogramm bei jedem Integrationsschritt aufgerufen wird, muss die Hilfsvariable irgendwo in der Eingabe noch ein zweites Mal auftauchen,* am besten unter `OUTPUTS`. Das Beispiel muss also um die Anweisungen

```
OUTPUTS
  PDUMM,....
```

ergänzt werden.

3.12 Die Vektornotation

SCEPTRE verwendet in den generierten FORTRAN-Unterprogrammen die gleichen Variablennamen wie sie in der Eingabe vom Benutzer gewählt wurden (z. B. PC, PL, C001 etc.). Werden mehr als 70 Elemente benutzt, schaltet SCEPTRE in eine Vektornotation um, d. h. alle Elemente werden in einem Vektor X und die definierten Parameter in einem Vektor P abgespeichert. Zusätzlich wird eine Korrespondenzliste ausgedruckt.

Eine Möglichkeit, Formelfunktionen zu verwenden, zeigt das folgende, allerdings nicht gute Beispiel:

```
ELEMENTS
  CX, .... = QC (VCX)
DEFINED PARAMETERS
  PL = 4711
FUNCTIONS
  QC(B) = (PL/(1.+B)*DLOG(2.+B))                    .
```

Die Kapazität CX wird mittels der Funktion QC aus den Parametern VCX und PL berechnet. Da PL nicht als Argument beim Formelaufruf auftritt, wird nach FORTRAN-Konvention der Wert von der lokalen Variablen PL eingesetzt. Solange SCEPTRE die Variablenbezeichnungen beibehält, d. h. bis zu 70 Elementen, ist der Parameter PL im FORTRAN-Unterprogramm bekannt und der Formelaufruf funktioniert. Schaltet SCEPTRE in die Vektornotation um, wird der Parameter PL in das Vektorelement z. B. P(2) umbenannt, in der Formelfunktion steht aber PL, d. h. der FORTRAN-Compiler setzt für PL Hausnummern ein. Man umgeht diese Gefahr, wenn man alle benötigten Parameter in der Funktion als Argumente übergibt.

[7] `REAL*8, COMPLEX*16`
[8] siehe Anhang B.1

```
ELEMENTS
  CX, .... = QC (PL,VCX)
FUNCTIONS
  QC(A,B) = (A/(1.+B)*DLOG(2.+B))
```

3.13 Wahl der Zustandsvariablen

SCEPTRE wählt die Zustandsvariablen nach eigenen Kriterien aus. Die Reihenfolge der Netzwerkbeschreibungen unter `ELEMENTS` spielt dabei eine gewisse Rolle. Durch Umsortierung der Eingabeanweisungen wählt SCEPTRE unter Umständen andere Zustandsvariablen aus. Zusätzlich sollte man unter `RUN CONTROLS` die Anweisung

```
NO ELEMENT SORT
```

einfügen. Von der Wahl der Zustandsvariablen und den Zeitkonstanten im Netzwerk hängt die Integrationsschrittweite ab. Unter Umständen können günstigere Integrationsschrittweiten erzielt werden. Bei umfangreichen Simulationen empfiehlt es sich daher — um Rechenzeit zu sparen — diese Methoden vorher auszuprobieren.

3.14 Berechnung von Integralen

Da die Ableitungen definierter Parameter spezifiziert werden können, gibt es eine einfache Möglichkeit, Integrale zu berechnen. Soll die Funktion $f(t)$ mit dem Anfangswert $f(0) = number$ integriert werden, lauten die Anweisungen unter `DEFINED PARAMETERS`

DP*name* = *f(t)*

P*name* = *number*

Am Ende der Simulation enthält `Pname` den Wert

$$\texttt{Pname} = \int_0^{\texttt{STOP TIME}} f(t)\,\mathrm{d}\,t + number.$$

Damit man die Integrale quantitativ einfacher auswerten kann, werden am Ende einer TR-Analyse der erste und letzte Wert jeder Ausgabevariablen ausgedruckt.

Hinweis: Als Anfangswert dürfen keine Variablen benutzt werden.

3.15 Simulationsstatistik

Am Ende einer AC- und TR-Analyse druckt SCEPTRE eine Statistik und einige Kontrollmeldungen aus.

```
INTEGRATION ROUTINE           ....
```

Es wird das verwendete Integrationsverfahren ausgegeben.

`CURRENT SIMULATION TIME ....`

Es wird die Zeit ausgedruckt, bis zu der die Simulation tatsächlich durchgeführt wurde.

`INTEGRATION STEP COUNTER ....`

Dieser Zähler gibt die Zahl der erfolgreichen Integrationsschritte an und entspricht somit der Anzahl der zur Verfügung stehenden Ergebnisse.

`INTEGRATION PASS COUNTER ....`

Dieser Zähler gibt die Anzahl der Integrationsschritte plus der Integrationsversuche an, d. h. er enthält auch die Integrationsschritte, die mit verkleinerten Schrittweiten wiederholt wurden.

`ELAPSED EXECUTION TIME [MIN] ....`

Diese Zeitangabe bezieht sich auf die Verweilzeit der Simulation im Rechner, sie ist sehr stark von der Rechnerauslastung abhängig.

`CPU EXECUTION TIME [SEC] ....`

Diese Zeitangabe entspricht der tatsächlichen Rechenzeit, jedoch nur für die Simulation. Die Rechenzeiten für das Verarbeiten der Eingabedaten, die FORTRAN-Übersetzung und das Linken sind nicht darin enthalten.

`TERMINATION CONDITION ....`

Diese Information ist die wichtigste und sollte deshalb nach jeder Simulation zuerst kontrolliert werden. Folgende Meldungen sind möglich:

- `STOP TIME EXCEEDED (NORMAL STOP)`
  Die Simulation ist normal verlaufen.
- `MAXIMUM INTEGRATION PASSES EXCEEDED`
  Die Simulation ist vorzeitig abgebrochen worden, da mehr Integrationsschritte notwendig waren, als vorgegeben.
- `SMALLER MINIMUM STEP SIZE REQUIRED`
  siehe Kapitel 3.8.
- `SOLUTION TIME LIMIT EXCEEDED`
  Die vorgegebene Rechenzeit ist aufgebraucht, die Simulation ist vorzeitig abgebrochen worden.
- `OPTIONAL TERMINATION CONDITION MET`
  Ein vom Benutzer vorgegebenes Abbruchkriterium ist erfüllt worden[9].

Bei der AC-Analyse enthält die Statistik folgende Angaben, die sich selbst erklären:

```
INITIAL FREQUENCY           ....
LAST CALCULATED FREQUENCY   ....
NUMBER FREQUENCY STEPS      ....
TYPE FREQUENCY RUN          ....
```

[9] siehe Kapitel 2.10.9: Ablaufsteuerung

3.16 Druckausgabe

Jede Variable, die man unter `OUTPUTS` angibt, kann auch tabellarisch ausgedruckt werden. Voreingestellt ist `MAXIMUM PRINT POINTS = 0`, d. h. der Ausdruck wird unterdrückt.

Benutzt man den Befehl

```
PRINT INTERVAL = number
```

dann wird diese Druckausgabe *zusätzlich* erzeugt. Auch hier kann man die Standard-Druckausgabe mit dem o. a. Kommando unterbinden.

Die Anweisungen

```
PLOT INTERVAL = ....
PRINT INTERVAL = ....
```

gehören zu der Überschrift `RUN CONTROLS` und nicht `OUTPUTS`.

3.17 Anfangsbedingungen

3.17.1 Arbeitspunkt

Zu Beginn jeder TR-Analyse haben die Spannungen an allen Elementen und die Ströme durch alle Elemente den Wert Null. Dies trifft auch auf Dioden-Stromquellen zu. Enthält eine Schaltung Energiespeicher und/oder Dioden-Stromquellen, stellt sich der Arbeitspunkt der Schaltung erst nach einer hinreichend langen Einschaltzeit der Versorgungsspannung ein.

SCEPTRE bietet die Möglichkeit, diesen Arbeitspunkt automatisch berechnen zu lassen mit der Anweisung `RUN INITIAL CONDITIONS [ONLY]` (siehe Seite 21 und Seite 24).

Vorsicht ist bei zeitabhängigen Tabellen geboten. Es werden grundsätzlich die Werte zur Zeit $t = 0$ eingesetzt. Liegt zu dieser Zeit eine Sprungfunktion vor, wird der erste Wert angenommen. Betrachtet man dazu die Funktion in Abb. 2.1 (Seite 20) und die dazugehörige Tabelle `TABLE XYZ7` (Abschnitt 2.9.2), würde SCEPTRE zum Zeitpunkt $t = 0$ den Wert 0 annehmen. Lässt man das erste Wertepaar (0,0) in der Tabelle weg, ergibt sich für die Anfangsbedingung der Wert 1. Der Spannungsverlauf ändert sich dadurch nicht.

Alternativ lassen sich die Anfangsbedingungen auch manuell unter `INITIAL CONDITIONS` vorgeben (siehe Seite 19).

Man kann auch beide Verfahren gleichzeitig anwenden. Die Werte unter `INITIAL CONDITIONS` werden dann als Startwerte für die Berechnung des Arbeitspunktes genommen.

Die Anfangsbedingungen lassen sich nicht berechnen, wenn Ströme durch Widerstände als unabhängige Variablen in Tabellen oder Gleichungen auftreten (statt dessen kann man I_x durch V_x/R_x ausdrücken, siehe dazu auch Abschnitt 3.9) oder wenn Größen in Abhängigkeit von Kapazitäten oder Induktivitäten verwendet werden, da diese bei der Berechnung der Anfangsbedingungen noch nicht existieren.

Achtung: enthält das zu untersuchende Problem Differentialgleichungen (`P…, DP…`), kann man die Anfangsbedingungen nur mittels der impliziten Integration bestimmen (siehe nächster Abschnitt).

3.17.2 Implizite Integration

Standardmäßig werden die Anfangsbedingungen nach der Newton-Raphson Methode iterativ berechnet. Unter Umständen konvergiert das Verfahren jedoch nicht oder das Netzwerk enthält für dieses Verfahren eine ungültige Topologie (E-L Maschen, J-C Knoten). In diesem Fall könnte man eine TR-Analyse mit „eingefrorenen“ Anregungsfunktionen durchführen und so lange warten, bis sich das Netzwerk auf die Anfangswerte eingependelt hat. Diese Methode ist nicht besonders effektiv, besonders dann, wenn die Zeitkonstanten im Netzwerk wertemäßig weit auseinander liegen.

Hier bietet SCEPTRE alternativ das implizite Integrationsverfahren an, das eine „Pseudo“-TR-Analyse mit folgenden Parametern durchführt:

```
MINIMUM STEP SIZE = 0
MAXIMUM STEP SIZE = 1E74 (oder 1E37)
STARTING STEP SIZE = 1E-8
```

Die „Integration”wird beendet, wenn für die Ableitungen des Zustandsvektors y gilt

$$|\dot{y}| \leq 10^{-8}|y| + 10^{-6} \ .$$

Um statt der Newton-Raphson Methode das implizite Verfahren zu aktivieren, müssen je nach Analyseart unter `RUN CONTROLS` folgende Anweisungen eingefügt werden:

- nur DC-Analyse (mit oder ohne Wiederholungsläufe)

```
RUN CONTROLS
    RUN INITIAL CONDITIONS ONLY
    RUN IC VIA IMPLICIT
```

- DC- und TR-Analyse

```
RUN CONTROLS
    RUN IC VIA IMPLICIT
    STOP TIME = ...
```

- DC- und TR-Analyse, Wiederholungsläufe nur mit TR-Analyse

```
RUN CONTROLS
    RUN IC VIA IMPLICIT
    STOP TIME = ...
RERUN DESCRIPTION
    ...
```

- DC- und TR-Analyse, Wiederholungsläufe ebenfalls mit DC- und TR-Analyse

```
RUN CONTROLS
    RUN IC VIA IMPLICIT
    STOP TIME = ...
RERUN DESCRIPTION
RUN CONTROLS
    RUN IC VIA IMPLICIT
    ...
```

3.17.3 Wiederholungsläufe

DC-Analyse

Die iterative Berechnung der Anfangswerte nach der Newton-Raphson Methode benutzt als Startwerte die unter `INITIAL CONDITIONS` angegebenen Strom- und Spannungswerte. Fehlende Werte werden zu Beginn der Iteration auf Null gesetzt. Die Vorgabe geeigneter Startwerte kann unter Umständen die Konvergenz beschleunigen und die Berechnung erheblich verkürzen.

Das gleiche Problem tritt bei Wiederholungsläufen auf. Es läßt sich Rechenzeit sparen, wenn man die Endwerte der vorherigen DC-Analyse als Startwerte für die weiteren DC-Analysen einsetzt. Bei mehr als einem Wiederholungslauf hat man die Wahl, welche Startwerte benutzt werden sollen: sollen die DC-Analysen aller Wiederholungsläufe mit den Endwerten der 1. DC-Analyse beginnen, lauten die Anweisungen *im 1. Lauf*:

```
RUN CONTROLS
    RUN INITIAL CONDITIONS ONLY
    (RUN IC VIA IMPLICIT)
    IC FOR RERUNS = MASTER RESULTS
RERUN DESCRIPTION (n)
    ...
```

Es können auch die Werte aus dem vorherigen Lauf übernommen werden:

```
RUN CONTROLS
    RUN INITIAL CONDITIONS ONLY
    (RUN IC VIA IMPLICIT)
    IC FOR RERUNS = PRECEDING RESULTS
RERUN DESCRIPTION (n)
    ...
```

Das implizite Verfahren ist optional (Klammern weglassen!).

TR-Analyse

Statt die Anfangsbedingungen bei Wiederholungsläufen über die DC-Analyse zu berechnen, kann man ebenso die Endwerte einer TR-Analyse als Startwerte weiterer TR-Analysen verwenden. Auch hier können die Startwerte entweder aus den Ergebnissen der 1. TR-Analyse (`IC FOR RERUN = MASTER RESULTS`) oder aus der vorherigen TR-Analyse (`IC FOR RERUNS = PRECEDING RESULTS`) eingesetzt werden.

3.18 Periodische Funktionen

Die Erzeugung von beliebigen, periodischen Funktionsverläufen kann man in SCEPTRE auf verschiedene Arten realisieren: durch arithmetische Anweisungen, Tabellen und eigene FORTRAN Unterprogramme. Dabei dürfen diese Funktionen auch springen, d. h. zu einem unabhängigen Wert können zwei abhängige Werte vorgegeben werden (z. B. ideale Rechteckfunktion).

3.18.1 Ideale Funktionen

Mit einer einfachen EXPRESSION-Anweisung lassen sich ideale, periodische Funktionen generieren. Haben diese Funktionen keine direkte Einwirkung auf die Netzwerkdynamik, z. B. als Referenzfunktion, können Ungenauigkeiten vor allem an den Sprungstellen auftreten, da sich die Integrationsschrittweite nur nach der Dynamik des Netzwerkes richtet und u. U. der Integrationszeitpunkt nicht mit dem Sprungzeitpunkt zusammenfällt.

Rechteckfunktion

Durch Kombination der Vorzeichen- und Sinus-/Kosinusfunktionen lassen sich ideale Rechteckimpulse erzeugen.
Beispiel: mit der Anweisung

```
EXXX, ....... = X1 (SIGN (311.,SIN (314.15*TIME) ) )
```

wird eine Rechteckspannung mit einer Frequenz von 50 Hz und einer Amplitude von ± 311 Einheiten erzeugt. Durch Hinzufügen eines Offsets und einer Phasenverschiebung lassen sich die Rechteckimpulse in weiten Grenzen den Erfordernissen anpassen.

Rampenfunktion

Beispiel: zur Erzeugung einer „Kippspannung“ kann folgende Anweisung benutzt werden:

```
ERAMP, ..... = X2 ( 3.8 + (8.2-3.8)*MOD ( (TIME/400E-6) ,1D0) )
```

Die Spannung setzt bei 3.8 ein und steigt innerhalb von 400E-6 Zeiteinheiten linear auf den Wert von 8.2. Danach wiederholt sich der gesamte Zyklus.

Getastete Sinusfunktion (Zeitmultiplex-Signal)

Abbildung 3.1 zeigt eine getastete Sinusfunktion, die mit folgender Anweisung erzeugt wurde:

```
..... = X12 (SIN(6.281*TIME)*(0.5+SIGN(0.5,SIN(75.500*TIME))))
```

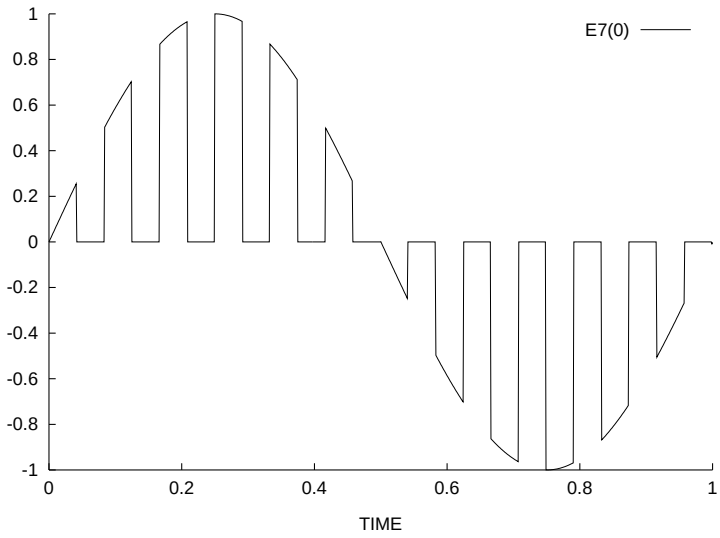


Abbildung 3.1: Zeitmultiplex-Signal

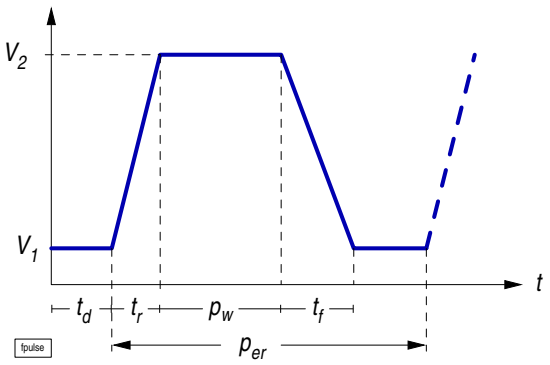


Abbildung 3.2: Die periodische Funktion FPULSE

3.18.2 FPULSE

Häufig werden Rechteckimpulse zum Testen oder Ansteuern von Schaltungen benutzt. Sprungstellen in den Funktionsverläufen verschlechtern im allgemeinen drastisch die Konvergenz des Integrationsverfahrens. Aus diesem Grund verwendet man statt der idealen Funktion vorteilhafter eine Trapezfunktion mit endlichen Anstiegs- und Abfallzeiten.

Die Funktion FPULSE erzeugt einen periodischen Impuls entsprechend Abbildung 3.2. Aufgerufen wird diese Funktion wie eine eigene FORTRAN Funktion:

```
.... = FPULSE ( v1, v2, td, tr, tf, pw, per )
```

.

Die Bedeutung der einzelnen Parameter geht aus der Abbildung 3.2 hervor. Der Parameter *td* stellt eine einmalige Verzögerungszeit dar.

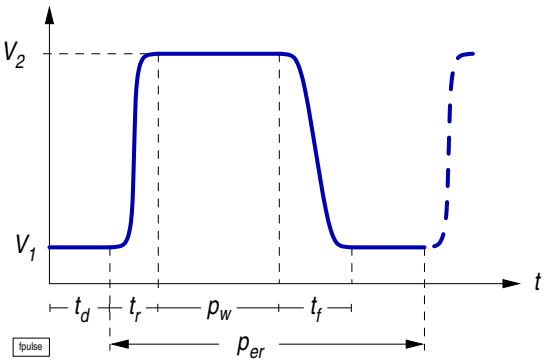


Abbildung 3.3: Die periodische Funktion FSINSQ

3.18.3 FSINSQ

Ersetzt man die Flanken eines idealen Rechteckimpulses durch $\sin^2$-Verläufe, erhält man die Funktion FSINSQ entsprechend der Abbildung 3.3.

```
.... = FSINSQ ( v1, v2, td, tr, tf, pw, per )
```

Die Bedeutung der Parameter geht aus der Abbildung 3.3 hervor. Der Parameter *td* stellt eine einmalige Verzögerungszeit dar.

3.19 Reservierte Variablennamen

Die vom Benutzer gewählten Namen können unter unglücklichen Umständen zu Konflikten mit SCEPTRE-internen Programmvariablen führen. Diese Probleme werden erst durch den FORTRAN-Compiler entdeckt.

Folgende Variablennamen sollte man vermeiden:

`PS` zusammen mit `DPS`.

SCEPTRE unterscheidet bei den freiwählbaren Namen zwischen Groß- und Kleinschreibung. Je nach FORTRAN-Compiler werden u. U. alle kleingeschriebenen Variablen in Großbuchstaben umgewandelt. Damit entfällt das Unterscheidungsmerkmal und es kommt zu Laufzeitfehlern.

Beispiel:

```
PMI = .....
Pmi = .....
```

Je nach FORTRAN-Compiler führen bestimmte Dummy-Argumente bei der `EQUATION`-Anweisung zu Übersetzungsfehlern (sie werden mit programminternen Variablen gleichen Namens verwechselt).

Beispiel:

```
Q9 (A,B,C,....) = .....
```

besser

```
Q9 (AA,BB,CC,...) = .....
```

3.20 Wechselstromquellen

Bei der Angabe von Wechselstromquellen (siehe Abschnitt 2.5.8) müssen rechts vom Gleichheitszeichen *zwei* Angaben, durch Komma getrennt, *in runden Klammern* angegeben werden (Abweichung vom Format für die TR-Analyse). Andernfalls wird die Quelle *ohne Fehlernachricht* ignoriert. Beispiel:

```
E1, Gnd - 2.2 = (325.27,0.)
```

4 Beispiele

4.1 A01: Inverter-Schaltung mit RC-Last

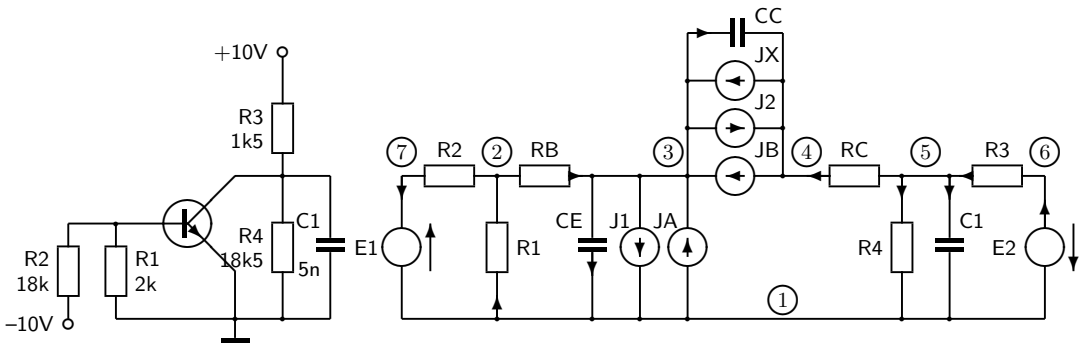


Abbildung 4.1: Inverterschaltung und für SCEPTRE aufbereitetes Ersatzschaltbild

Abb. 4.1 zeigt eine einfache Inverterschaltung, die mit einem RC Netzwerk belastet wird, sowie die für SCEPTRE aufbereitete Ersatzschaltung. Es soll der Einfluss von γ-Strahlung auf die Funktion der Schaltung untersucht werden.

Der durch die Strahlung ausgelöste Ladungstransport im Halbleiter wird durch eine zeitabhängige Stromquelle `JX` simuliert. Der Stromverlauf in Abhänigkeit von der Zeit wird durch `TABLE 1` beschrieben. Die Anfangsbedingungen (Arbeitspunkt) werden explizit vorgegeben. Es handelt sich hier um eine reine Zeitanalyse.

Für den Transistor wird die Ersatzschaltung nach Ebers-Moll benutzt. Die Beschreibung der beiden Dioden `J1` und `J2` erfolgt durch spannungsgesteuerte Stromquellen mit der in SCEPTRE eingebauten Funktion `DIODE EQUATION (x1, x2)`. Dabei errechnet sich der Diodenstrom `J` direkt aus der Diodenspannung `VJ` nach der Beziehung $J = x_1(\exp(x_2 VJ) - 1)$.

Um den Einfluß verschiedener Stromverläufe von `JX` beurteilen zu können, wird die Schaltung in zwei Wiederholungsläufen mit veränderter Stromverläufen der Stromquelle `JX` (`TABLE 1`) untersucht. Abb. 4.2 und 4.3 zeigen einen Teil der Ergebnisse.

```
CIRCUIT DESCRIPTION
A01: INVERTER CIRCUIT LOADED WITH RC NETWORK
  ELEMENTS
    E1, 7-1 = 10
    E2, 1-6 = 10
    CE, 3-1 = EQUATION 1 (5., 70., J1)
    CC, 3-4 = EQUATION 1 (8., 370., J2)
    C1, 5-1 = 500
    R1, 1-2 = 2
    R2, 2-7 = 17
```

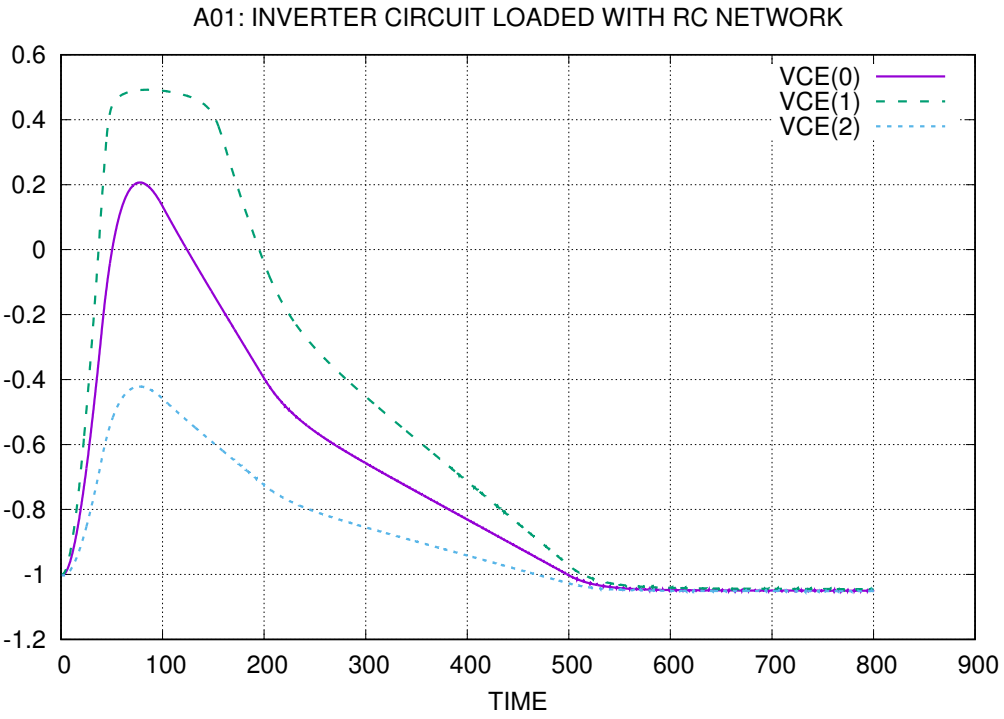


Abbildung 4.2: Spannung `VCE` in Abhängigkeit von `JX` entsprechend `TABLE 1`

```
    R3, 6-5 = 1.5
    R4, 5-1 = 18.5
    RB, 2-3 = .3
    RC, 5-4 = .015
    J1, 3-1 = DIODE EQUATION (1.E-7, 35.)
    J2, 3-4 = DIODE EQUATION (5.E-7, 37.)
    JA, 1-3 = .1 * J2
    JB, 4-3 = .98 * J1
    JX, 4-3 = TABLE 1 (TIME)
  OUTPUTS
    VCE, VCC, VC1, IR3, J1, PLOT
  INITIAL CONDITIONS
    VC1 = 9.25
    VCE = -1
    VCC = -10.25
  FUNCTIONS
    TABLE 1
      0,   0
      40,   .8
      100,  .5
      200,  .25
      500, 0
      600, 0
    EQUATION 1 (A,B,C) = (A + B * C)
  RUN CONTROLS
    MAXIMUM PRINT POINTS = 0
    INTEGRATION ROUTINE = TRAP
    STOP TIME = 800
RERUN DESCRIPTION (2)
```

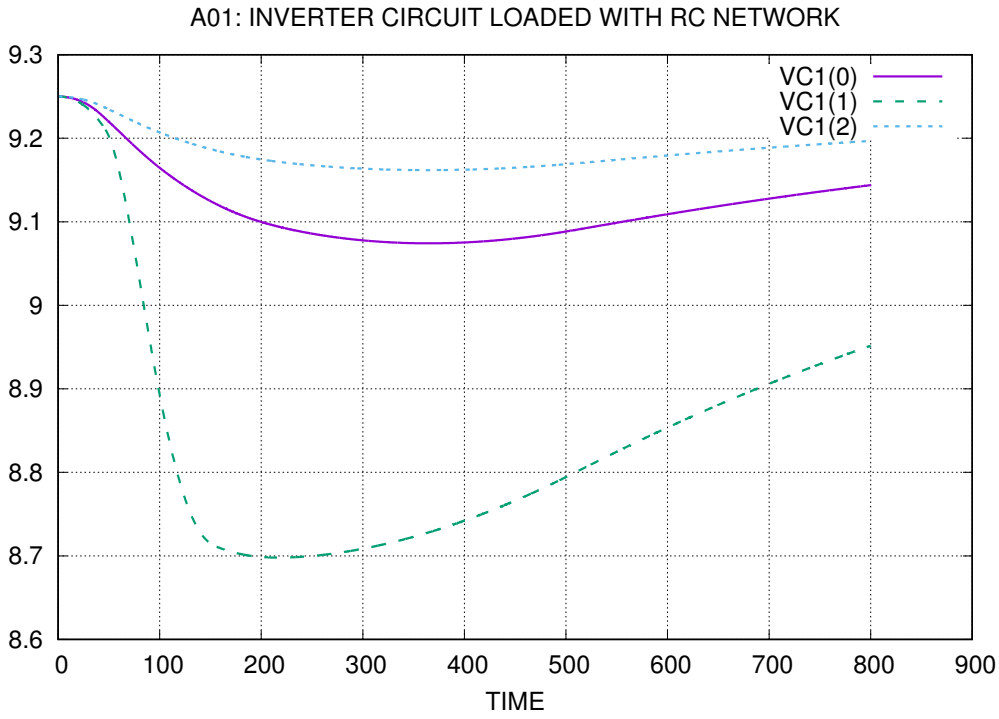


Abbildung 4.3: Spannung `VC1` in Abhängigkeit von `JX` entsprechend `TABLE 1`

```
  FUNCTIONS
    TABLE 1 = 0,   0,      0
              40,  1.2,     .4
             100,   .75,    .25
             200,   .375,   .125
             500,  0,      0
             600,  0,      0
END
```

4.2 A02: Emitterfolger mit Übertrager

Eine kombinierte Kollektor- und Emitterschaltung, die einen mit einem Widerstand belasteten Übertrager treibt, zeigt Abb. 4.4. Die Schaltung wird durch eine rampenförmige Spannung `E1` angesteuert. Das für die Simulation ausreichende Transistormodell ist ebenfalls wiedergegeben. Es wird für beide Transistoren verwendet und permanent gespeichert. Die Strom-Spannungscharakteristiken der beiden Dioden `J1` und `J2` werden durch die Diodentabellen 1 und 2 wiedergegeben. Der Transistor `T2` arbeitet mit einer geringeren Stromverstärkung als `T1` (0.975 statt 0.98). Der Übertrager besteht aus den Spulen `L1` und `L2`, die über die Gegeninduktivität `M` miteinander gekoppelt sind.

Die Simulation soll an einer betriebsbereiten Schaltung durchgeführt werden, d. h. es interessiert hier nicht der Einschwingvorgang, den das Einschalten der Spannungsversorgung verursacht. Deshalb muss vorher der Arbeitspunkt der Schaltung bestimmt werden mit (`RUN INITIAL CONDITIONS`), der dann als Ausgangszustand für die nachfolgende Transient-Analyse dient (`STOP TIME`).

Die Spannungsverläufe `VL1` und `VL2` sind in Abbildung 4.5 wiedergegeben.

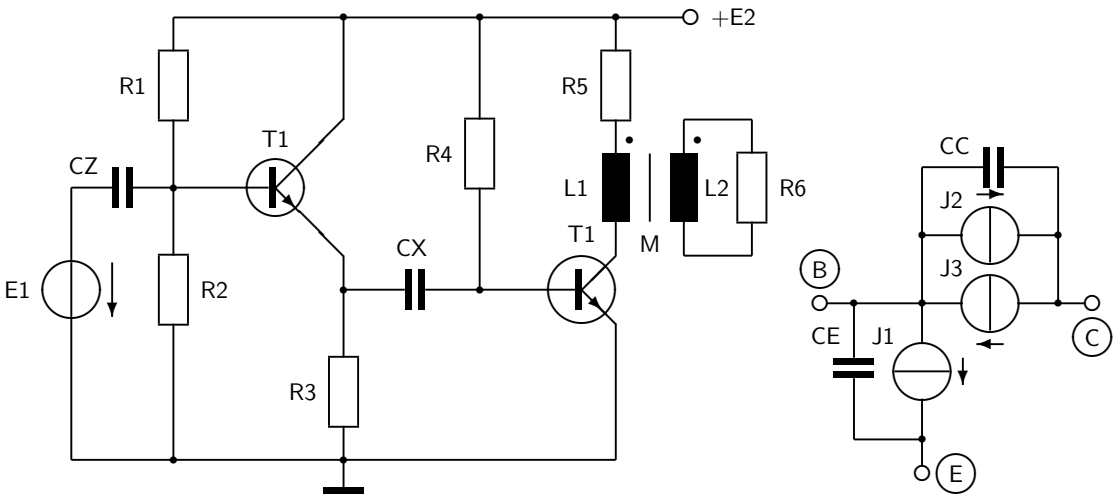


Abbildung 4.4: Emitterfolger mitÜbertrager, Transistor-Ersatzschaltbild

```
MODEL DESCRIPTION
MODEL 2N9999AA (PERM) (B-E-C)
ELEMENTS
  CE,B-E=EQUATION 1(5.,40.,TABLE 1(VCE))
  CC,B-C=EQUATION 1(10.,400.,TABLE 2(VCC))
  J1,B-E=DIODE TABLE 1
  J2,B-C=DIODE TABLE 2
  J3,C-B=P1*J1
DEFINED PARAMETERS
  P1=.98
FUNCTIONS
  EQUATION 1(A,B,C)=(A+B*C)
  DIODE TABLE 1
    0,0,.3,0,.65,.05,.7,.6,.72,1.4,.73,2,.74,3.4,.77,10,.8,22
  DIODE TABLE 2
    0,0,.58,0,.62,.4,.64,1,.66,2,.67,3,.69,7,.7,12
CIRCUIT DESCRIPTION
EXAMPLE 2: TRANSFORMER COUPLED AMPLIFIER
ELEMENTS
  E1,1-2=TABLE 1(TIME)
  DERIVATIVE E1=TABLE DE1
  E2,1-4=20
  CZ,2-3=1E3
  CX,5-6=1E3
  R1,4-3=30
  R2,3-1=20
  R3,5-1=2
  R4,4-6=240
  R5,4-7=3.3
  R6,9-1=1.8
  T1,3-5-4=MODEL 2N9999AA (TEMP)
  T2,6-1-8=MODEL 2N9999AA (TEMP,CHANGE P1=.975)
  L1,7-8=100
  L2,9-1=900
  M,L1-L2=299.7
OUTPUTS
```

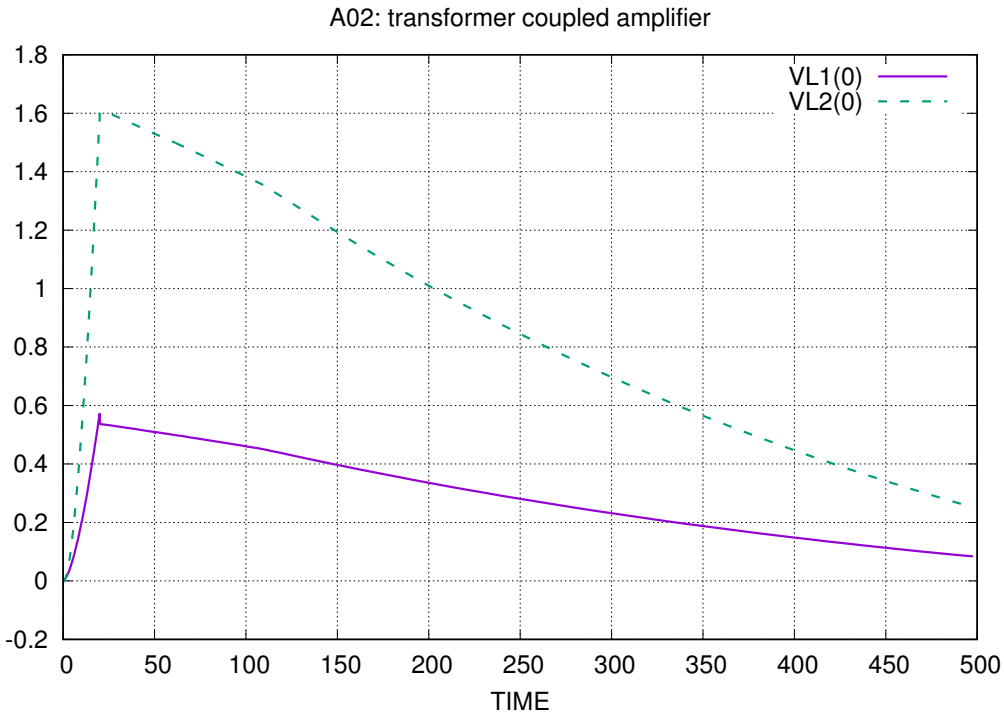


Abbildung 4.5: Spannungen `VL1` und `VL2` am Übertrager

```
  VR6,VL1,VL2,PLOT
FUNCTIONS
  TABLE 1
    0,0,50,.5,100,.5
  TABLE DE1
    0,.01,50,.01,50,0,100,0
RUN CONTROLS
  STOP TIME=500
  RUN INITIAL CONDITIONS
  INTEGRATION ROUTINE=IMPLICIT
  MAXIMUM PRINT POINTS=21
END
```

4.3 A03: Darlington-Schaltung (DC-Analyse)

Das Prinzip einer Darlington Schaltung geht aus Abb. 4.6 hervor. Gesucht sind die Spannungen, die sich an den Widerständen `R4` und `R5` einstellen, sowie die Leistungsaufnahme der Schaltung, bei unterschiedlicher Stromverstärkung des Transistors `T1`. Dazu genügt eine Gleichstromanalyse (`RUN INITIAL CONDITIONS ONLY`). Die Beschreibung des Modells für den Transistor `2N706A` existiert schon in der Modellbibliothek und erscheint deshalb nicht in den SCEPTRE-Eingabedaten. Sie ist jedoch der Übersicht wegen aufgelistet.

```
MODEL 2N706A (PERM) (B-E-C)
  ELEMENTS
    CE, 1-E = Q1(5.,70.,J1)
    CC, 1-2 = Q1(8.,370.,J2)
```

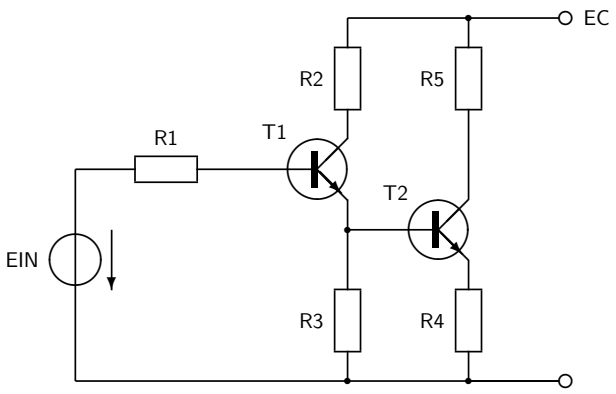


Abbildung 4.6: Darlington Schaltung

```
    RB, B-1 = .3
    RC, C-2 = .015
    J1, 1-E = DIODE EQUATION(1.E-7,35.)
    J2, 1-2 = DIODE EQUATION(5.E-7,37.)
    JA, E-1 = .1*J2
    JB, 2-1 = P1*J1
    JX, 2-1 = 0
  DEFINED PARAMETERS
    P1 = 0.98
  OUTPUTS
    VCE,VCC,J1,PLOT
  FUNCTIONS
    Q1(A,B,C) = (A+B*C)
```

Die Stromverstärkung wird durch den Parameter `P1` im Modell `2N706A` dargestellt. Der Bezug zu der Stromverstärkung des Transistors `T1` erfolgt durch Anhängen des Bezugsnamens `T1` an den internen Modellnamen, also `P1T1`.

```
CIRCUIT DESCRIPTION
Example 3: Darlington pair
  ELEMENTS
    EC  , 1-6   = 10
    EIN , 1-2   = 1
    R1  , 2-3   = 20
    R2  , 6-5   = 5
    R3  , 4-1   = 200
    R4  , 8-1   = .4
    R5  , 6-7   = 1
    T1  , 3-4-5 = MODEL 2N706A (PERM)
    T2  , 4-8-7 = MODEL 2N706A (PERM)
  DEFINED PARAMETERS
    PEC = X1 (EC*IEC)
    PEIN= X2 (EIN*IEIN)
  OUTPUTS
    PEC,PEIN,VR3,VR4
  RUN CONTROLS
    RUN INITIAL CONDITIONS ONLY
RERUN DESCRIPTION (3)
```

```
  DEFINED PARAMETERS
    P1T1= .96, .93, .9
END
```

Die Ausgabe der Ergebnisse setzt sich aus den unter `OUTPUTS` angegebenen Größen zusammen, und zwar sowohl von der Hauptschaltung als auch von der Modellbeschreibung.

```
I/C TRANSIENT VALUES AT TIME EQUALS ZERO:
VCET1 = 3.3818865E-01
VCCT1 =-8.9376695E+00
VCET2 = 4.4221916E-01
VCCT2 =-8.8226781E+00

RESULTS OF INITIAL CONDITION COMPUTATIONS
PEC           5.3007344E+00
PEIN          2.7598821E-04
VR3           6.5620879E-01
VR4           2.1082735E-01

I/C TRANSIENT VALUES AT TIME EQUALS ZERO:
VCET1 = 3.3769191E-01
VCCT1 =-8.9456210E+00
VCET2 = 4.4163421E-01
VCCT2 =-8.8381601E+00

RERUN      1
RESULTS OF INITIAL CONDITION COMPUTATIONS
PEC           5.1910074E+00
PEIN          5.4289718E-04
VR3           6.5128728E-01
VR4           2.0655488E-01

I/C TRANSIENT VALUES AT TIME EQUALS ZERO:
VCET1 = 3.3696571E-01
VCCT1 =-8.9570427E+00
VCET2 = 4.4077602E-01
VCCT2 =-8.8603292E+00

RERUN      2
RESULTS OF INITIAL CONDITION COMPUTATIONS
PEC           5.0340188E+00
PEIN          9.2654405E-04
VR3           6.4422545E-01
VR4           2.0044292E-01

I/C TRANSIENT VALUES AT TIME EQUALS ZERO:
VCET1 = 3.3626101E-01
VCCT1 =-8.9679012E+00
VCET2 = 4.3993970E-01
VCCT2 =-8.8813264E+00

RERUN      3
RESULTS OF INITIAL CONDITION COMPUTATIONS
PEC           4.8854795E+00
PEIN          1.2915639E-03
```

```
VR3             6.3752024E-01
VR4             1.9466077E-01

SIMULATION PROGRAM HAS TERMINATED
```

4.4 A04: Kleinsignal-Ersatzschaltung

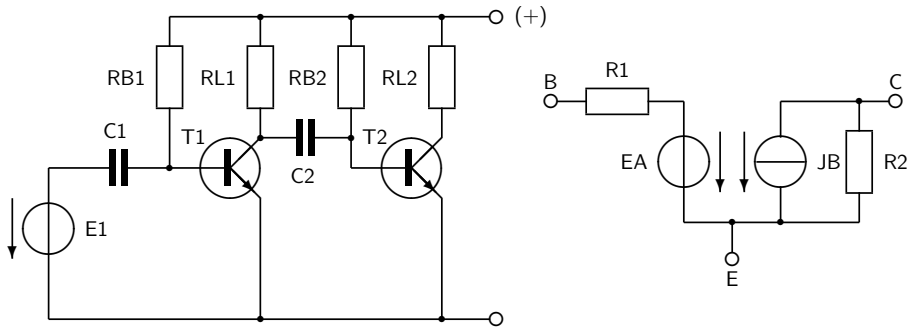


Abbildung 4.7: Kleinsignal-Schaltung, Ersatzschaltbild mit h-Parametern

Dieses Beispiel soll den Einsatz von SCEPTRE bei der Analyse einer Kleinsignal-Schaltung nach Abbildung 4.7 zeigen. Es handelt sich dabei um einen RC gekoppelten, linearen, zweistufigen Verstärker, dessen Verhalten bei einer sinusförmigen Eingangsspannung `E1` von 1 mV und 100 kHz untersucht werden soll. Für die Transistoren wird als temporäres Modell das Ersatzschaltbild mit h-Parametern benutzt. Da nur das Wechselstromverhalten interessiert, schließt man die Gleichspannungsquelle kurz. Die Simulation wird entweder nach 30000 ns beendet oder wenn die Ausgangsspannung am Lastwiderstand `RL2` größer als 20 V wird.

```
MODEL DESCRIPTION
  MODEL SS1 (TEMP) (B-E-C)
    ELEMENTS
      EA, E-X = .0005 * VR2
      R2, C-E = 2000
      R1, B-X = .3
      JB, C-E = 50. * IR1
CIRCUIT DESCRIPTION
A04 : small signal equivalent circuit
  ELEMENTS
    E1, 1-2 = X1(.001*DSIN(.000628*TIME))
    C1, 2-3 = 5E6
    C2, 4-5 = 5E6
    RB1,3-1 = 100
    RL1,1-4 = 1
    RB2,5-1 = 100
    RL2,1-6 = 1
    T1, 3-1-4 = MODEL SS1
    T2, 5-1-6 = MODEL SS1
  OUTPUTS
    VRL1, VRL2, VC1, VC2, PLOT
  RUN CONTROLS
    STOP TIME = 30000
    INTEGRATION ROUTINE = TRAP
```

```
    MAXIMUM PRINT POINTS = 21
    TERMINATE IF (VRL2 .GE. 20.)
END
```

Der Verlauf der Spannungen an den Widerständen `RL1` und `RL2` ist in den Abbildungen 4.8 und 4.9 zu sehen.

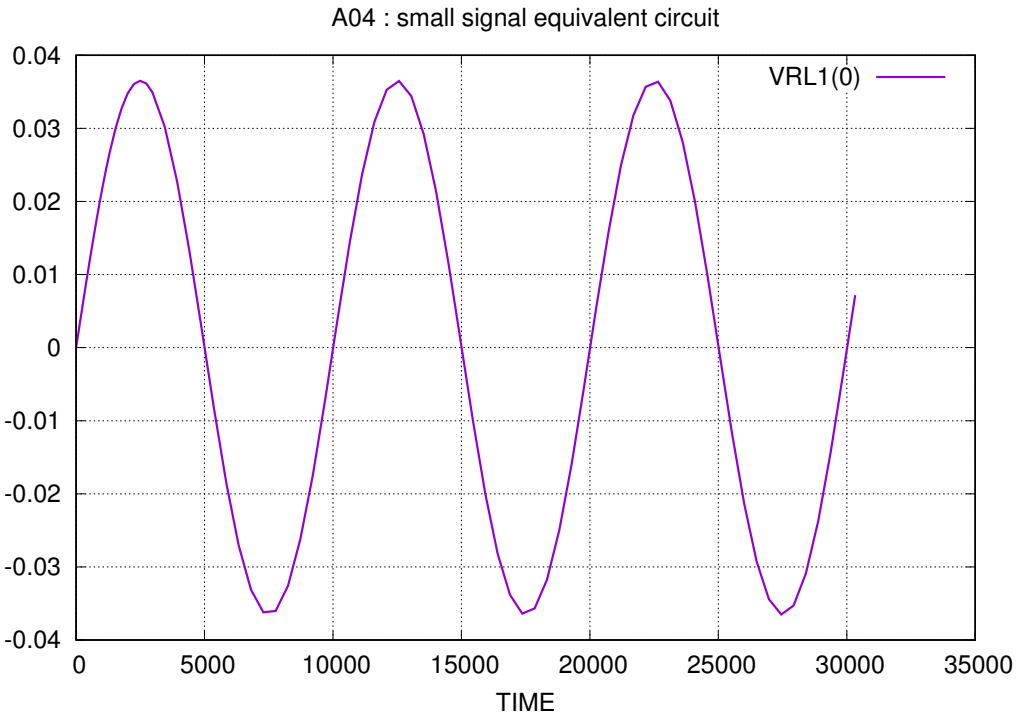


Abbildung 4.8: Spannungsverlauf am Widerstand `RL1`

4.5 A05: Differentialgleichungssystem

Dieses Beispiel zeigt die Anwendung der definierten Parameter zur Lösung eines Systems von Differentialgleichungen erster Ordnung. Dieses Gleichungssystem ist vollkommen unabhängig von einem elektrischen Netzwerk.

$$\begin{aligned} \dot{x} &= -6x + 5y + 10 \\ \dot{y} &= 5x - 7y + 2z \\ \dot{z} &= 0.2y - 0.2z - 0.5 \end{aligned}$$

Die Anfangsbedingungen lauten $x(0) = 6, y(0) = 5, z(0) = 4$ und werden durch die Parameter `PX, PY, PZ` gesetzt. Nach jedem Integrationsschritt werden ihnen neue Werte zugewiesen (Abb. 4.10).

```
CIRCUIT DESCRIPTION
Example A05: Solution of simultaneous differential equations
DEFINED PARAMETERS
  DPX = EQUATION 1 (PX, PY)
  DPY = EQUATION 2 (PX, PY, PZ)
```

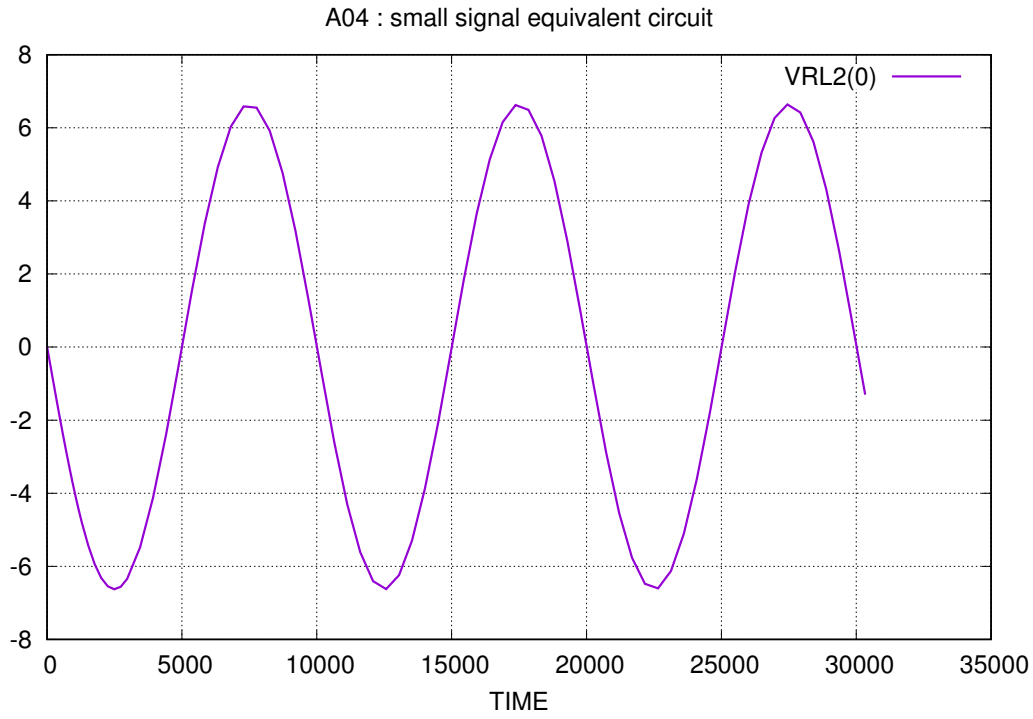


Abbildung 4.9: Spannungsverlauf am Widerstand RL2

```
  DPZ = EQUATION 3 (PY, PZ)
  PX  = 6
  PY  = 5
  PZ  = 4
OUTPUTS
  PX(X), PY(Y), PZ(Z), XSTPSZ, PLOT
FUNCTIONS
  EQUATION 1 (A,B) = (-6.*A+5.*B+10.)
  EQUATION 2 (A,B,C) = (5.*A-7.*B+2.*C)
  EQUATION 3 (A,B) = (.2*A-.2*B-.5)
RUN CONTROLS
  INTEGRATION ROUTINE = TRAP
  STOP TIME = 100
  MAXIMUM PRINT POINTS = 21
END
```

4.6 Toleranzanalysen einer Darlington-Schaltung

Die folgenden Beispiele befassen sich mit Toleranzanalysen der Darlington-Schaltung aus dem Beispiel A03 (siehe Abschnitt 4.3). Obwohl alle Toleranzanalysen in einem Simulationslauf durchgeführt werden können, werden sie hier einzeln vorgestellt.

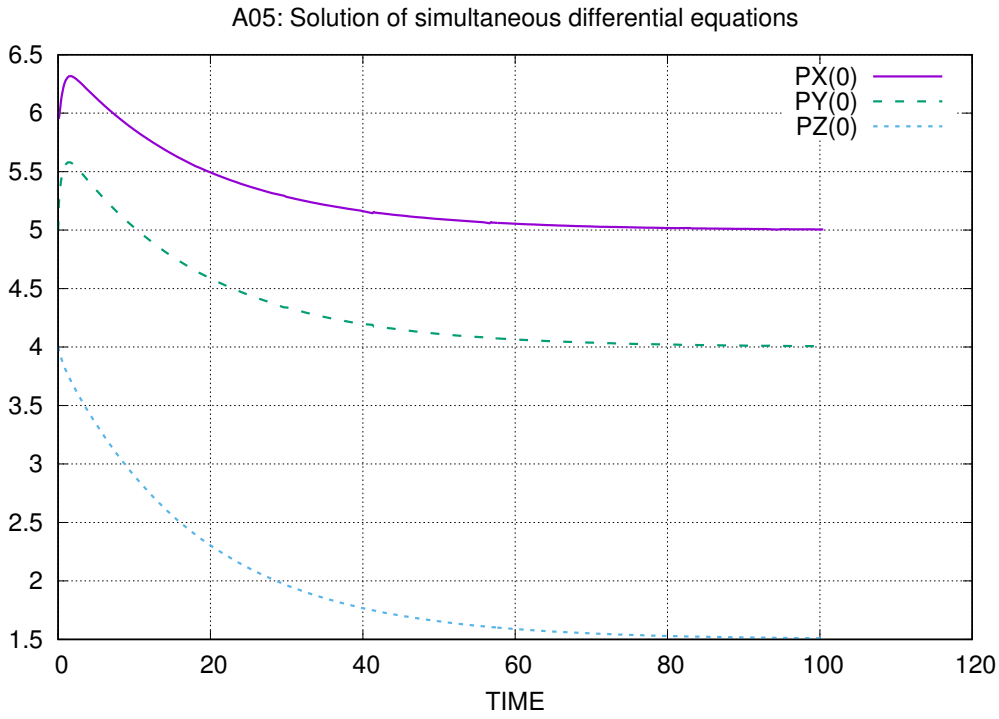


Abbildung 4.10: Verlauf von `PX, PY, PZ`

4.6.1 A07: Monte-Carlo-Analyse

Als unabhängige Variable für die Monte-Carlo-Analyse wird der Stromverstärkungsfaktor `P1` im Modell 2N706A gewählt, dessen Nominalwert bei 0.96 liegt, aber Schwankungen von 0.9 bis 0.98 aufweisen soll. Unter der Überschrift `MONTE CARLO` werden alle zu beobachtenden Variablen angegeben (vor dem Schrägstrich), gefolgt von den Parametern, die entsprechend ihrer Toleranzgrenzen variiert werden sollen (`P1T1, P1T2`). Es wird eine Gauß-Verteilung angenommen (Voreinstellung).

```
MODEL DESCRIPTION
MODEL 2N706A (TEMP) (B-E-C)
  ELEMENTS
    CE, 1-E = Q1(5.,70.,J1)
    CC, 1-2 = Q1(8.,370.,J2)
    RB, B-1 = .3
    RC, C-2 = .015
    J1, 1-E = DIODE EQUATION(1.E-7,35.)
    J2, 1-2 = DIODE EQUATION(5.E-7,37.)
    JA, E-1 = .1*J2
    JB, 2-1 = P1*J1
    JX, 2-1 = 0
  DEFINED PARAMETERS
    P1 = 0.96 (0.98, 0.9)
  FUNCTIONS
    Q1(A,B,C) = (A+B*C)
  OUTPUTS
    VCE, VCC, J1
CIRCUIT DESCRIPTION
A07: Darlington pair, MONTE CARLO
  ELEMENTS
```

```
    EC  , 1-6   = 10
    EIN , 1-2   = 1
    R1  , 2-3   = 20
    R2  , 6-5   = 5
    R3  , 4-1   = 200
    R4  , 8-1   = .4
    R5  , 6-7   = 1
    T1  , 3-4-5 = MODEL 2N706A
    T2  , 4-8-7 = MODEL 2N706A
  MONTE CARLO
    (VCET1,VCCT1,J1T1,VCET2,VCCT2,
     J1T2,PEC,PEIN,VR3,VR4/P1T1,P1T2)
  DEFINED PARAMETERS
    PEC = X1 (EC*IEC)
    PEIN= X2 (EIN*IEIN)
  OUTPUTS
    PEC,PEIN,VR3,VR4
  RUN CONTROLS
    RUN MONTE CARLO = 10
    RUN INITIAL CONDITIONS ONLY
END
```

Man beachte, wie man durch Anhängen des Modellbezugnamens jedes Element und jede Variable in der Gesamtschaltung ansprechen kann. Das Ergebnis der statistischen Analyse liefert

```
I/C TRANSIENT VALUES AT TIME EQUALS ZERO:

VCET1 = 3.5123970E-01
VCCT1 =-8.9126370E+00
VCET2 = 4.3874672E-01
VCCT2 =-8.9197511E+00

INPUTS TO MONTE CARLO

NAME       NOMINAL          LBOUND           UBOUND           MEAN             SIGMA

P1T1      9.6000000E-01    9.0000000E-01    9.8000000E-01    9.4000000E-01    1.3333333E-02
P1T2      9.6000000E-01    9.0000000E-01    9.8000000E-01    9.4000000E-01    1.3333333E-02

INITIAL RANDOM NUMBER =    127263527
DISTRIBUTION IS GAUSSIAN
NORMAL MONTE CARLO TERMINATION AFTER   10 ITERATIONS
FINAL RANDOM NUMBER =   1007121511

OBSERVED STATISTICS

INDEPENDENT      NOMINAL          MINIMUM          MAXIMUM          SAMPLE           SAMPLE
 VARIABLE         VALUE            VALUE            VALUE            MEAN             SIGMA

 P1T1          9.6000000E-01    9.3082453E-01    9.7906312E-01    9.5388296E-01    1.6480857E-02
 P1T2          9.6000000E-01    9.1926269E-01    9.5900296E-01    9.4255822E-01    1.1320930E-02


 DEPENDENT       NOMINAL          MINIMUM          MAXIMUM          SAMPLE           SAMPLE
 VARIABLE         VALUE            VALUE            VALUE            MEAN             SIGMA

 VCET1         3.5123970E-01    3.5109792E-01    3.6510355E-01    3.5782344E-01    4.0157705E-03
 VCCT1        -8.9126370E+00   -8.9210128E+00   -8.8596399E+00   -8.8927857E+00    2.2013686E-02
 J1T1          2.1824751E-02    2.1716713E-02    3.5455596E-02    2.7726820E-02    3.9307541E-03
```

```
 VCET2          4.3874672E-01    4.3440367E-01    4.3795162E-01    4.3649493E-01    1.4850834E-03
 VCCT2         -8.9197511E+00   -9.0300613E+00   -8.9392260E+00   -8.9773819E+00    3.5350405E-02
 J1T2           4.6675032E-01    4.0093037E-01    4.5394055E-01    4.3189852E-01    2.2252503E-02
 PEC            4.6903307E+00    4.0210736E+00    4.5616323E+00    4.3372422E+00    2.2675435E-01
 PEIN           8.7254006E-04    6.3257750E-04    1.8368401E-03    1.2575918E-03    4.3076374E-04
 VR3            6.3104773E-01    6.0275509E-01    6.2660595E-01    6.1664745E-01    9.7793613E-03
 VR4            1.8670015E-01    1.6037217E-01    1.8157624E-01    1.7275943E-01    8.9010012E-03

RESULTS OF INITIAL CONDITION COMPUTATIONS

VCET1           3.6510355E-01
VCCT1          -8.8596399E+00
J1T1            3.5455596E-02
VCET2           4.3445810E-01
VCCT2          -9.0300613E+00
J1T2            4.0169492E-01
PEC             4.0323994E+00
PEIN            1.4793579E-03
VR3             6.0486548E-01
VR4             1.6067799E-01
```

4.6.2 A08: Empfindlichkeitsanalyse

Die Empfindlichkeitsanalyse dient dazu, den Einfluss einzelner (unabhängiger) Elemente und Parameter auf andere (abhängige) Netzwerkvariablen — ausgedrückt durch partielle Ableitungen — abschätzen zu können. Für die Darlington-Schaltung werden zwei Sätze von partiellen Ableitungen gesucht, die unter der Überschrift `SENSITIVITY` angegeben werden. Toleranzangaben sind nicht erforderlich. Während SCEPTRE für die abhängigen Netzwerkelemente die partiellen Ableitungen selber berechnen kann, muss der Benutzer für eine definierte, abhängige Funktion `Pname` das vollständige Differential `GPname` vorgeben.

```
MODEL DESCRIPTION
MODEL 2N706A (TEMP) (B-E-C)
  ELEMENTS
    CE, 1-E = Q1(5.,70.,J1)
    CC, 1-2 = Q1(8.,370.,J2)
    RB, B-1 = .3
    RC, C-2 = .015
    J1, 1-E = DIODE EQUATION(1.E-7,35.)
    J2, 1-2 = DIODE EQUATION(5.E-7,37.)
    JA, E-1 = .1*J2
    JB, 2-1 = P1*J1
    JX, 2-1 = 0
  DEFINED PARAMETERS
    P1 = 0.96
  FUNCTIONS
    Q1(A,B,C) = (A+B*C)
  OUTPUTS
    VCE, VCC, J1
CIRCUIT DESCRIPTION
Example A08: Darlington pair, SENSITIVITY
  ELEMENTS
    EC  , 1-6   = 10
    EIN , 1-2   = 1
    R1  , 2-3   = 20
    R2  , 6-5   = 5
```

```
    R3  , 4-1   = 200
    R4  , 8-1   = .4
    R5  , 6-7   = 1
    T1  , 3-4-5 = MODEL 2N706A
    T2  , 4-8-7 = MODEL 2N706A
  SENSITIVITY
    (VCET1,VCCT1,J1T1,PR3,PRR3/P1T1,P1T2,R3)
    (VCET2,VCCT2,PEIN,VR4/P1T1,P1T2,R3,R4)
  DEFINED PARAMETERS
    PEIN= X2 (EIN*IEIN)
    GPEIN=P4*DIEIN
    P4  = X4 (EIN)
    PR3  = XR3 (R3*IR3*IR3)
    GPR3 = PDR3*DR3+PDIR3*DIR3
    PDR3 = XDR3 (IR3*IR3)
    PDIR3= XDIR3 (2.*IR3*R3)
    PRR3 = XRR3 (IR3*VR3)
    GPRR3= PDIRR3*DIR3+PDVRR3*DVR3
    PDIRR3 = XDIRR3 (VR3)
    PDVRR3 = XDVRR3 (IR3)
  OUTPUTS
    VR3,VR4
  RUN CONTROLS
    RUN SENSITIVITY
    RUN INITIAL CONDITIONS ONLY
END
```

Bei der vorliegenden Schaltung soll der Einfluss auf die Eingangsleistung `PEIN` und der Verlustleistung `PR3` untersucht werden. Dabei wird die Leistung im Widerstand `R3` sowohl über den Spannungsabfall `VR3` als auch über den Strom `IR3` berechnet. Aus

$$P_{EIN} = I_{EIN} E_{IN}$$

folgt für das vollständige Differential

$$dP_{EIN} = \frac{\partial P_{EIN}}{\partial I_{EIN}} dI_{EIN} + \frac{\partial P_{EIN}}{\partial E_{IN}} dE_{IN}.$$

Da $E_{IN} = \text{const}$ und nicht zu den unabhängigen Variablen gehört, ergibt sich in der SCEPTRE-Schreibweise:

```
PEIN = X1 (EIN * IEIN)
P2   = X2 (EIN)
GPEIN= P2 * DIEIN
```

Für die Leistung PR3 erhält man

$$P_{R3} = I_{R3}^2 R_3$$

$$dP_{R3} = \frac{\partial P_{R3}}{\partial I_{R3}} dI_{R3} + \frac{\partial P_{R3}}{\partial R_3} dR_3$$

und in der SCEPTRE-Schreibweise

```
PR3  = X3 (R3 * IR3 * IR3)
P4   = X4 (2. * IR3 * R3)
P5   = X5 (IR3 * IR3)
GPR3 = P4 * DIR3 + P5 * DR3
```

Bezieht man die Leistung auf den Spannungsabfall `VR3`, so ergibt sich

$$P_{RR3} = I_{R3} V_{R3}$$

$$dP_{RR3} = \frac{\partial P_{RR3}}{\partial I_{R3}} dI_{R3} + \frac{\partial P_{RR3}}{\partial V_{R3}} dV_{R3}$$

und in der SCEPTRE-Schreibweise

```
PRR3  = X6 (IR3 * VR3)
P7    = X7 (VR3)
P8    = X8 (IR3)
GPRR3 = P7 * DIR3 + P8 * DVR3
```

Wie man aus den Ergebnissen der Empfindlichkeitsanalyse erkennt, ergeben sich für die Leistungen `PR3` und `PRR3` die gleichen partiellen Ableitungen bzw. Empfindlichkeiten.[1]

```
SENSITIVITY CALCULATION(S)
```

SET NUMBER	DEPENDENT VARIABLE	INDEPENDENT VARIABLE	PARTIAL DERIVATIVE	NORMALIZED SENSITIVITY
1.	VCET1	P1T1	2.23659125E-01	6.11299797E-01
		P1T2	-4.07175175E-02	-1.11288150E-01
		R3	-3.75988248E-06	-2.14092110E-03
	VCCT1	P1T1	1.23628942E+00	-1.33163490E-01
		P1T2	-1.24486175E-01	1.34086834E-02
		R3	-1.14951356E-05	2.57951394E-04
	J1T1	P1T1	1.70846451E-01	7.51498104E+00
		P1T2	-3.11028820E-02	-1.36811486E+00
		R3	-2.87206067E-06	-2.63192979E-02
	PR3	P1T1	2.07019909E-01	9.98134165E+01
		P1T2	5.27852161E-01	2.54500777E+02
		R3	5.68080410E-06	5.70617891E-01
	PRR3	P1T1	1.03764443E-01	5.00294086E+01
		P1T2	2.64134241E-01	1.27350752E+02
		R3	-2.11814175E-06	-2.12760299E-01
2.	VCET2	P1T1	1.04334237E-02	2.28288587E-02
		P1T2	2.66473139E-02	5.83056702E-02
		R3	7.88042611E-07	3.59224392E-04
		R4	-1.11083071E-02	-1.01273074E-02
	VCCT2	P1T1	2.44690518E-01	-2.63351403E-02
		P1T2	1.09869936E+00	-1.18248970E-01
		R3	1.84816183E-05	-4.14397623E-04
		R4	2.06232105E-01	-9.24833451E-03

[1] Die Ergebnisse stimmen nicht mit der Original IBM-Version überein!

```
            PEIN       P1T1       -1.49908934E-02      -1.64935209E+01
                       P1T2       -1.24411528E-03      -1.36882044E+00
                       R3         -1.14882427E-07      -2.63328717E-02
                       R4         -7.18255438E-03      -3.29271044E+00

           VR4         P1T1        6.81772684E-02       3.50563074E-01
                       P1T2        1.74127029E-01       8.95349841E-01
                       R3          5.14946906E-06       5.51629887E-03
                       R4          3.94163071E-01       8.44483683E-01

RESULTS OF INITIAL CONDITION COMPUTATIONS

VCET1         3.5123970E-01
VCCT1        -8.9126370E+00
J1T1          2.1824751E-02
VCET2         4.3874672E-01
VCCT2        -8.9197511E+00
J1T2          4.6675032E-01
VR3           6.3104773E-01
VR4           1.8670015E-01
```

4.6.3 A09: 'Worst Case'-Analyse

Aufgabe der Worst-Case-Simulation ist es, Extremwerte einer Zielfunktion (=abhängige Netzwerkvariable) zu finden für Werte einer oder mehrerer unabhängiger Netzwerkvariablen, die Toleranzen haben müssen.

```
MODEL DESCRIPTION
MODEL 2N706A (TEMP) (B-E-C)
  ELEMENTS
    CE, 1-E = Q1(5.,70.,J1)
    CC, 1-2 = Q1(8.,370.,J2)
    RB, B-1 = .3
    RC, C-2 = .015
    J1, 1-E = DIODE EQUATION(1.E-7,35.)
    J2, 1-2 = DIODE EQUATION(5.E-7,37.)
    JA, E-1 = .1*J2
    JB, 2-1 = P1*J1
  DEFINED PARAMETERS
    P1 = 0.96 (0.98, 0.9)
  FUNCTIONS
    Q1(A,B,C) = (A+B*C)
  OUTPUTS
    VCE, VCC, J1
CIRCUIT DESCRIPTION
Example A09: Darlington pair, WORST CASE
  ELEMENTS
    EC  , 1-6   = 10
    EIN , 1-2   = 1
    R1  , 2-3   = 20
    R2  , 6-5   = 5
    R3  , 4-1   = 200 (195,205)
    R4  , 8-1   = .4 (0.35,0.45)
    R5  , 6-7   = 1
    T1  , 3-4-5 = MODEL 2N706A
```

```
    T2  , 4-8-7 = MODEL 2N706A
  WORST CASE
    (VCET1,VCCT1,J1T1,PEIN/P1T1,P1T2,R3)
    (VCET2,VR4/P1T2,R3,R4)
  DEFINED PARAMETERS
    PEIN= X2 (EIN*IEIN)
    GPEIN=P4*DIEIN
    P4 = X4 (EIN)
  OUTPUTS
    VR3,VR4
  RUN CONTROLS
    RUN WORST CASE
    RUN INITIAL CONDITIONS ONLY
END
```

Für die Zielfunktion von PEIN muss das vollständige Differential vorgegeben werden.

Es folgt ein Ausschnitt aus den Ergebnissen für die Extremwerte von VCCT1 mit P1T1, P1T2, R3 als unabhängige Parameter. Für die Nominalwerte von P1T1, P1T2 und R3 beträgt die Spannung VCCT1=-8.9126. Sie nimmt ihren größten Wert von -8.894 an, wenn P1T1=0.98, P1T2=0.9579 und R3=200 betragen. Der kleinste Wert für VCCT1=-8.955 ergibt sich für P1T1=0.9, P1T2=0.966 und R3=200.

```
WORST_CASE_COMPUTATION -  NOMINAL VALUE

OBJECTIVE FUNCTION VCCT1   = -8.91263697E+00

INDEPENDENT VARIABLE        VALUE        GRADIENT COMPONENT      LOWER BOUND         UPPER BOUND

       P1T1             9.60000000E-01      1.23628942E+00      9.00000000E-01      9.80000000E-01
       P1T2             9.60000000E-01     -1.24486175E-01      9.00000000E-01      9.80000000E-01
       R3               2.00000000E+02     -1.14951356E-05      1.95000000E+02      2.05000000E+02



     LOW VALUE LOCATED AT DISTANCE -6.03034076E-02 ALONG GRADIENT = LOWER BOUND  9.00000000E-01 OF
      INDEPENDENT VARIABLE P1T1

    HIGH VALUE LOCATED AT DISTANCE  2.01011359E-02 ALONG GRADIENT = UPPER BOUND  9.80000000E-01 OF
     INDEPENDENT VARIABLE P1T1



WORST CASE COMPUTATION -     HIGH VALUE

OBJECTIVE FUNCTION VCCT1   = -8.89478115E+00

INDEPENDENT VARIABLE        VALUE        GRADIENT COMPONENT

       P1T1             9.80000000E-01      1.60698069E+00
       P1T2             9.57986132E-01     -1.76735527E-01
       R3               2.00000000E+02     -1.61045665E-05


WORST CASE COMPUTATION -      LOW VALUE

OBJECTIVE FUNCTION VCCT1   = -8.95520062E+00
```

```
INDEPENDENT VARIABLE        VALUE        GRADIENT COMPONENT

       P1T1             9.00000000E-01      6.78924171E-01
       P1T2             9.66041604E-01     -4.67066273E-02
       R3               2.00000001E+02     -4.44565760E-06
```

4.6.4 A10: Optimierung

In diesem Beispiel werden die Zielfunktionen von PEC und VR3 bezüglich der unabhängigen Variablen P1 und EC, die mit Toleranzgrenzen versehen sein müssen, minimiert.

```
MODEL DESCRIPTION
MODEL 2N706A (TEMP) (B-E-C)
  ELEMENTS
    CE, 1-E = Q1(5.,70.,J1)
    CC, 1-2 = Q1(8.,370.,J2)
    RB, B-1 = .3
    RC, C-2 = .015
    J1, 1-E = DIODE EQUATION(1.E-7,35.)
    J2, 1-2 = DIODE EQUATION(5.E-7,37.)
    JA, E-1 = .1*J2
    JB, 2-1 = P1*J1
  DEFINED PARAMETERS
    P1 = 0.96 (0.98, 0.9)
  FUNCTIONS
    Q1(A,B,C) = (A+B*C)
  OUTPUTS
    VCE, VCC, J1
CIRCUIT DESCRIPTION
Example A10: Darlington pair, OPTIMIZATION
  ELEMENTS
    EC  , 1-6   = 10 (9.5, 10.5)
    EIN , 1-2   = 1
    R1  , 2-3   = 20
    R2  , 6-5   = 5
    R3  , 4-1   = 200
    R4  , 8-1   = .4
    R5  , 6-7   = 1
    T1  , 3-4-5 = MODEL 2N706A
    T2  , 4-8-7 = MODEL 2N706A
  OPTIMIZATION
    (VR3,PEC/P1T1,P1T2,EC)
  DEFINED PARAMETERS
    PEC = XEC (EC*IEC)
    GPEC= P3*DIEC+P5*DEC
    P3 = X3 (EC)
    P5 = X5 (IEC)
  OUTPUTS
    VR3,VR4
  RUN CONTROLS
    RUN OPTIMIZATION = 40
    RUN INITIAL CONDITIONS ONLY
END
```

Bei diesem Beispiel treten definierte Parameter sowohl als abhängige (`PEC`) als auch unabhängige Variablen (`P1`) auf. Für die abhängige Variable muss das vollständige Differential vorgegeben werden. Als unabhängige Variablen dürfen definierte Parameter nur dann verwendet werden, wenn sie Quellen erster und zweiter Ordnung beschreiben. Es folgt ein Teil der Optimierungsergebnisse. Erklärungen über die Bedeutung der in den Ergebnissen vorkommenden Parametern findet man in den entsprechenden Kapiteln von [2, 3].

```
INITIAL VALUES OF OPTIMIZATION PARAMETERS:

OBJECTIVE FUNCTION                  VR3
NUMBER OF INDEPENDENT VARIABLES       3
NUMBER OF RANDOM STEPS                0
INITIAL H MATRIX FACTOR            1.00000000E+00
CONVERGENCE CRITERION              1.00000000E-07
RANDOM STEP SIZE CONTROL           2.00000000E-01
MINIMUM FUNCTION ESTIMATE          0.00000000E+00
H MATRIX DETERMINANT               1.00000000E+00

INITIAL VALUES OF INDEPENDENT VARIABLES:

              NOMINAL            LOWER             UPPER
VARIABLE       VALUE             BOUND             BOUND

P1T1        9.60000000E-01    9.00000000E-01    9.80000000E-01
P1T2        9.60000000E-01    9.00000000E-01    9.80000000E-01
EC          1.00000000E+01    9.50000000E+00    1.05000000E+01
INITIAL APPROXIMATION TO H MATRIX
     1.0000000E+00  0.0000000E+00  0.0000000E+00
     0.0000000E+00  1.0000000E+00  0.0000000E+00
     0.0000000E+00  0.0000000E+00  1.0000000E+00

OPTIMIZATION RESULTS:

TERMINATION CONDITION                       OPTIMIZATION RUN COMPLETED
OBJECTIVE FUNCTION                          VR3
VALUE OF OBJECTIVE FUNCTION AT TERMINATION   5.65021325E-01
OPTIMIZATION PASS COUNTER                     13
MAXIMUM PASSES SPECIFIED                      40

INDEPENDENT VARIABLE VALUES AT TERMINATION:
                                  GRADIENT        TRANSFORMED       TRANSFORMED
VARIABLE       VALUE              COMPONENT      VALUE (RADIANS)     GRADIENT

P1T1        9.00000002E-01    4.50846113E-01   -2.86201981E-04   -5.16132196E-06
P1T2        9.00000004E-01    5.48451029E-01   -4.44554674E-04   -9.75265843E-06
EC          1.00000000E+01    1.28031073-147    1.57079633E+00    6.40155365-148
```

4.7 A11: Kleinsignal-Ersatzschaltung (AC)

Abbildung 4.11 zeigt eine Transistorschaltung, deren Frequenzverhalten in der Umgebung ihres Arbeitspunktes untersucht werden soll.

```
MODEL DESCRIPTION
  MODEL 101M (B-C-E)
    ELEMENTS
      RBC,  BR-B  = 50.
      RC,   C -BR = 2.4E9
```

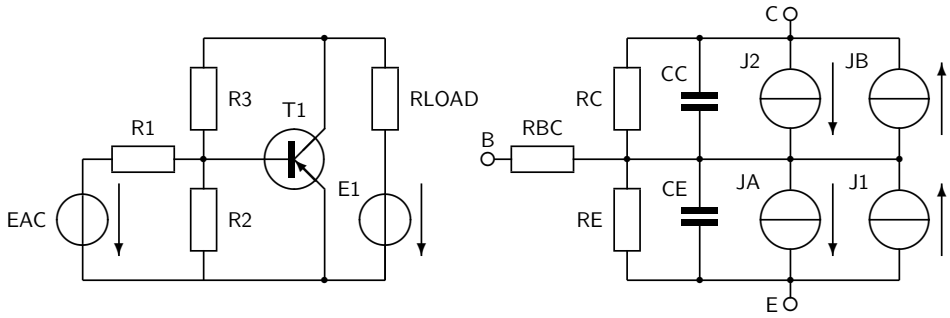


Abbildung 4.11: Kleinsignalschaltung, Transistor-Ersatzschaltbild

```
      RE,   E -BR = 5.7E8
      CC,   C -BR = Q1 (5.5E-12, .8, VCC, .3, 2.88E-7, J2, 3.12E-12)
      CE,   E -BR = Q2 (3.5E-12, 17E-9, J1)
      JA,   BR-E  = .332 * J2
      JB,   BR-C  = .986 * J1
      J1,   E -BR = DIODE EQUATION (60E-9, 38.4)
      J2,   C -BR = DIODE EQUATION (54.3E-8, 38.4)
    FUNCTIONS
      Q1 (A,B,C,D,E,F,G) = (A/(B-C)**D + E*(F+G))
      Q2 (A,B,C)         = (A+B*C)
    OUTPUTS
      VCC, VCE, NYQUIST, PLOT
      CC, CE
      JA, JB, DEGREES
      J1, J2, RADIANS, PLOT
CIRCUIT DESCRIPTION
Example A11: small signal AC-Analysis
             around a circuit's DC operating point
  ELEMENTS
    EAC,  1 - B = (0.2, 0.)
    R1,   B - 2 = 25E4
    R2,   2 - 1 = 2000
    R3,   2 - 3 = 11500
    RLOAD,3 - 4 = 100
    E1,   1 - 4 = -20
    T1,   2-3-1 = MODEL 101M
  OUTPUTS
    VR2, VR3, COMPLEX, PLOT
  RUN CONTROLS
    RUN INITIAL CONDITIONS
    PRINT EIGENVALUES
    PRINT EIGENVECTORS
    RUN AC
    INITIAL FREQUENCY = 1E5
    FINAL FREQUENCY = 1E6
    NUMBER FREQUENCY STEPS = 20
    TYPE FREQUENCY RUN = LINEAR
END
```

Es wird eine kombinierte DC-/AC-Analyse durchgeführt. Dabei ist folgendes zu beachten: *bei der DC-Analyse werden alle AC-Quellen Null gesetzt. Umgekehrt werden bei der AC-Analyse alle DC-Quellen*

Null gesetzt. Aus diesem Grund sollte man AC-Quellen von DC-Quellen mittels einer Kapazität oder eines großen Widerstandes (hier: R1) entkoppeln. Außerdem werden bei der AC-Analyse die Diodenquellen durch entsprechende Widerstände, die dem Arbeitspunkt des Transistors entsprechen, ersetzt.

Teilausdruck der Simulationsergebnisse:

```
OUTPUTS BEFORE AC RUN

VCCT1        -1.3023927E+01
VCET1         3.6223429E-01
CCT1          2.3449600E-12
CET1          1.1243506E-09
JAT1         -1.8027600E-07
JBT1          6.5009336E-02
J1T1          6.5932389E-02
J2T1         -5.4300000E-07
VR2          -4.0836859E-01
VR3           1.2977793E+01

 J9 SOURCE 'J1T1    ' IS REPLACED BY RESISTOR= 0.3949750
 J9 SOURCE 'J2T1    ' IS REPLACED BY RESISTOR= 0.2400000E+13

LSNEW: COMPLEX EIGENVALUES

    I     REAL              IMAGINARY       MAGNITUDE

    1   -0.65292E+10      0.00000E+00      0.65292E+10
    2   -0.33353E+08      0.00000E+00      0.33353E+08

ATRACE =  -0.6562516388467601E+10

EIGENVALUE SUM =  -0.6562516388467599E+10  0.0000000000000000E+00

WHEN COMPARING A*S WITH S*LAMBDA, THE WORST ERROR OBTAINED WAS
(-0.19073E-05, 0.00000E+00) WHEN MULTIPLYING MODAL MATRIX BY
INVERSE MODAL MATRIX

  WORST ERROR IN   DIAGONAL   TERM WAS ( 0.00000E+00, 0.00000E+00)
  WORST ERROR IN OFF-DIAGONAL TERM WAS ( 0.00000E+00, 0.00000E+00)

SINEW: MODAL MATRIX ... S

                    1                              2
   1     1.0000E+00     0.0000E+00     9.9999E-01     0.0000E+00
   2    -2.0843E-03     0.0000E+00     3.9932E-03     0.0000E+00

SINEW: INVERSE MODAL MATRIX ...SI

                    1                              2
   1     6.5705E-01     0.0000E+00    -1.6454E+02     0.0000E+00
   2     3.4296E-01     0.0000E+00     1.6454E+02     0.0000E+00
```

Abbildungen 4.12 zeigt die die komplexe Spannung `VCCT1` als Funktion der Frequenz (Nyquist-Plot).

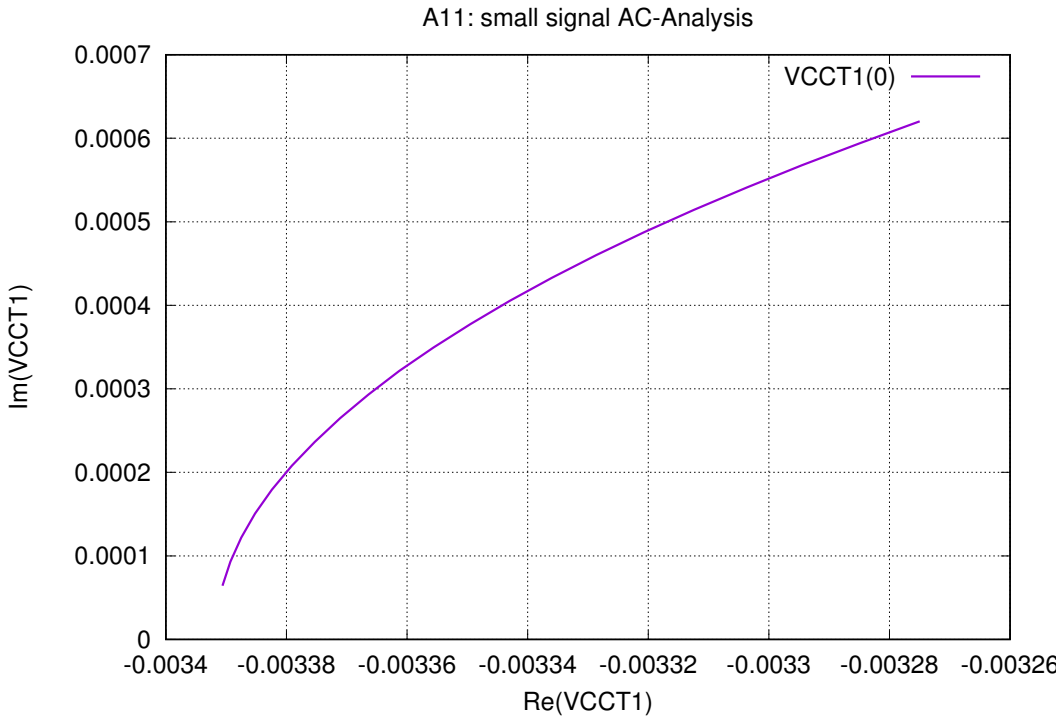


Abbildung 4.12: Nyquist-Plot von `VCCT1`

4.8 A13: Elektro-hydraulischer Stellantrieb

Die Schaltung nach Abb. 4.13 stellt einen elektro-hydraulischen Stellantrieb dar [4, Kapitel 9.7][2].

Die Führungsgröße wird durch die Spannungsquelle `EC=TABLE C (TIME)` vorgegeben. Die Regeldifferenz wird mit Hilfe des Operationsverstärkers und der beiden Transistoren verstärkt und wirkt über einen Magnetanker auf ein Servoventil. Die Regelabweichung wird durch einen induktiven Geber in eine Regelspannung umgewandelt, demoduliert und dem Regler mit umgekehrten Vorzeichen mit Hilfe der gesteuerten Spannungsquelle `EA` wieder zugeführt.

Die Größe `PDIS` stellt die Regelgröße dar, deren Verlauf in Abbildung 4.15 zu sehen ist. Der Zusammenhang zwischen dem Spulenstrom `ILTORK` und `PDIS` wird durch eine Übertragungsfunktion beschrieben und kann in SCEPTRE direkt mit Hilfe der definierten Parameter und ihrer Ableitungen (`PX1, PX2, DPX1, DPX2`) berechnet werden.

Durch die Verwendung von Nullstromquellen lassen sich Spannungen zwischen beliebigen Knotenpunkten verlust- und rückwirkungsfrei messen. Mit der Nullstromquelle `J57` erhält man z. B. das Differenzsignal `VJ57` am Eingang des Operationsverstärkers.

```
MODEL DESCRIPTION
MODEL HA 2700 (N-I-OUT-GND)
ELEMENTS
  EA1,    GND-3  = PAO*VRIN
  EA3,    GND-12 = TABLE 1 (VCA2)
  CIN,      N-I  = 1.E-12
  RIN,      N-I  = 1.E6
  CA1,      4-6  = 2280.E-6
```

[2] Autoren des Kapitels 9 sind A. Brent White und Michael A. Chipman, AFWL Albuquerque, New Mexico

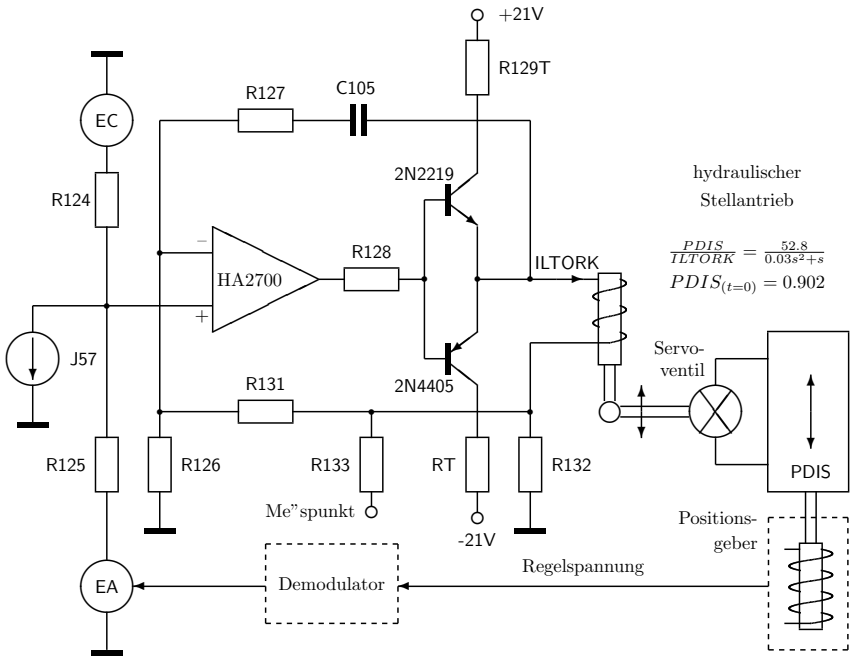


Abbildung 4.13: Schaltung des Regelkreises

```
  CA2,     5-GND= 45.E-12
  RA1,     3-4  = 100.
  RA2,     4-5  = 100.E3
  RA6,    12-OUT= 100.
  RA7,     6-GND= .028
DEFINED PARAMETERS
  PAO = 2.E5
FUNCTIONS
TABLE 1
  -20,-13, -13,-13, 13,13, 20,13

MODEL 2N2219 (B-E-C)
ELEMENTS
  CC, B1-C1 = 13.E-12
  CE, B1-E  = 22.E-12
  RB,  B-B1 = 50.
  RC,  C-C1 = 2.8
  JE, B1-E  = DIODE EQUATION (3.02E-14,40.)
  JC, B1-C1 = DIODE EQUATION (1.19E-13,38.)
  JN, C1-B1 = PAO*JE
  JI,  E-B1 = PAI*JC
DEFINED PARAMETERS
  PAO =.9927
  PAI = XAI (0.702*PAO)

MODEL 2N4405 (B-E-C)
ELEMENTS
  CE,  E-B1 = 55E-12
  CC, C1-B1 = 43.E-12
```

```
  RB, B1-B  = 20.
  RC, C1-C  = 1.4
  JE,  E-B1 = DIODE EQUATION (1.02E-13,39.)
  JC, C1-B1 = DIODE EQUATION (1.71E-13,38.)
  JN, B1-C1 = PAO*JE
  JI, B1-E  = PAI*JC
DEFINED PARAMETERS
  PAO = .9914
  PAI = XAI (0.6829*PAO)

CIRCUIT DESCRIPTION
A13: torquer closed-loop step response test circuit
     parameters for a 2% step increase from 88% point in displacement schedule
     Jensen, McNamee: Handbook of Circuit Analysis Languages and Techniques
     simplified, without radiation effects, etc.
     basis units: ohms, volts, farads henries amps, seconds
ELEMENTS
  J57,  57-0  = 0
  JT,   73-0  = 0
  JDR,  72-0  = 0
  JR1,  73-82 = 0
  JR2,  83-0  = 0.
  JOT,  71-0  = 0
  O3,57-70-71-0 = MODEL HA 2700
  EC,    0-56 = TABLE C (TIME)
  R124, 56-57 = 23.7E3
  R125, 57-11 = 20000
  EA,    0-11 = XA (PDIS*1.843)
  R126, 70-0  = 10E3
  C105, 73-81 = 4700E-12
  R127, 81-70 = 800E3
  R128, 71-72 = 511
  R129T,80-79 = 95.
  R131, 70-82 = 66.5E3
  R132, 82-0  = 10.
  R133, 83-82 = 10E3
  RTORK,73-TI = 50
  LTORK,82-TI = .2
  CTORK,73-82 = .127E-6
  T4,72-73-74 = MODEL 2N4405
  T3,72-73-79 = MODEL 2N2219
  E13,   0-80 = 21
  E14,  78-0  = 21.
  RT,   74-78 = 60.
DEFINED PARAMETERS
  PX1  = .902
  PX2  = 0.
  DPX1 = PX2
  DPX2 = XXX ((-1./.03)*PX2+1760.*ILTORK)
  PDIS = PX1
  PSTP = XSTP (LOG10(XSTPSZ))
INITIAL CONDITIONS
  ILTORK = 0
  VCCT3  = -2.1E1
  VCCT4  = -2.09162E1
  VCTORK = 0
```

```
OUTPUTS
  VJ57,EA,EC,VJOT,VJDR,VJT,ILTORK,PDIS,VJR2,VJR1,PSTP
FUNCTIONS
  TABLE C
  0,-1.97, .01,-1.97, .01,-2.07, 1.,-2.07
RUN CONTROLS
  X PLOT DIMENSION = 0
  INTEGRATION ROUTINE = IMPLICIT
  STOP TIME = .45
  MINIMUM STEPSIZE = 1E-20
END
```

Dieses interdisziplinäre Beispiel zeigt, wie man mit SCEPTRE elektrische und mechanische Größen sowie Übertragungsfunktionen miteinander verknüpfen kann. In [4, Kapitel 9.7] wird dieses Beispiel ausführlich erläutert.

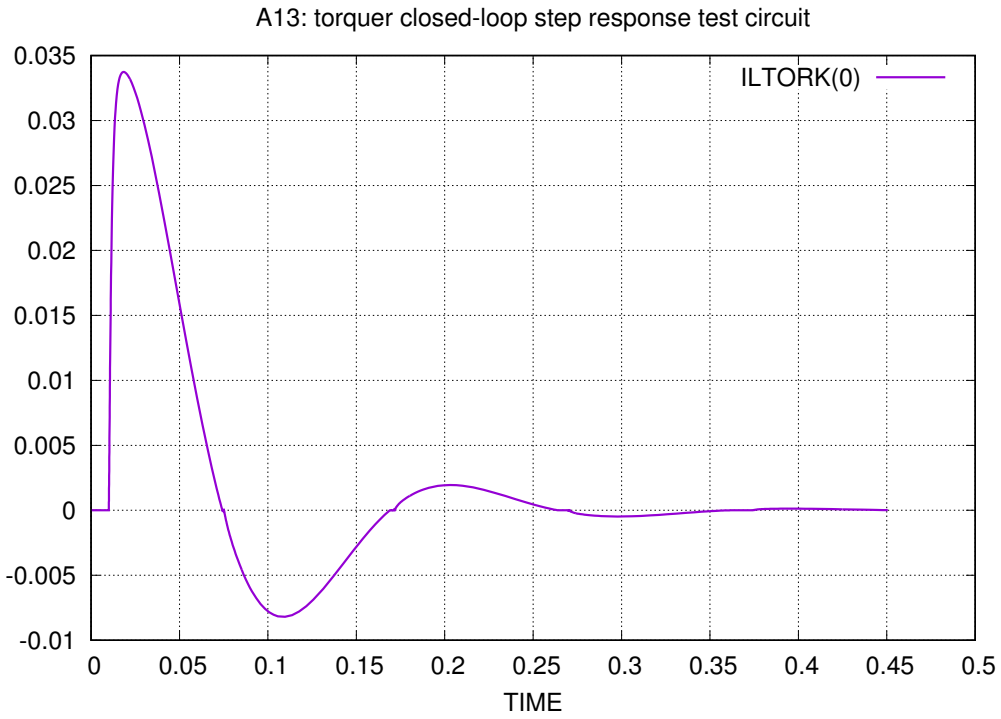


Abbildung 4.14: Magnetstrom `ILTORK` als Funktion der Zeit



Abbildung 4.15: Regelgröße `PDIS` als Funktion der Zeit

Teil II

Mathematische Beschreibung (Volume II)

5 Die Theorie hinter SCEPTRE

Dieser Teil des Benutzerhandbuchs beschreibt die mathematischen Algorithmen, die SCEPTRE verwendet. Im Wesentlichen stellen die folgenden Kapitel eine Übersetzung des Original-Handbuchs Volume II [3] dar. Die rechnerspezifischen Abläufe werden nicht behandelt.

5.1 Überblick

Jedes Programm zur Untersuchung des Zeitverhaltens eines dynamischen Systems dient dazu, die Programmierung der mathematischen Formulierung und des Differentialgleichungssystems zu übernehmen.

5.2 Genereller Lösungsansatz

SCEPTRE beeinhaltet 3 verschiedene Lösungsmethoden zur Berechnung eines vorgegebenen Netzwerkes:

1. Die Gleichstromanalyse (DC) ermittelt die Netzwerkspannungen als Anfangsbedingung für eine sich anschließende Transient- oder Wechselstromanalyse. Dabei wird nicht die Zeit als unabhängige Variable sondern als konstant angenommen. Die Ergebnisse der Gleichstromanalyse können für sich genommen werden oder gehen automatisch als Anfangswert in die sich anschließende Transient- oder Wechselstromanalyse ein. Somit dienen diese Ergebnisse als Anfangsbedingungen für ein Differentialgleichungssystem, welches von der Transient- bzw. Wechselstromanalyse gelöst wird.

2. Bei der Transientanalyse (TR) wird die Zeit als unabhängige Variable genommen und ein Differentialgleichungssystem als Funktion der Zeit gelöst. Man erhält als Ergebnis das Zeitverhalten eines vorgegebenen Netzwerkes. Wie bereits erwähnt, kann das Transientenprogramm in Verbindung mit der Gleichstromanalyse eingesetzt werden oder auch alleine, falls die Anfangsbedingungen des Netzwerkes bereits bekannt sind.

3. Die Wechselstromanalyse (AC) untersucht das Frequenzverhalten eines Netzwerkes. Dabei wird die Frequenz als unabhängige Variable genommen und das Gleichungssystem als Funktion der Frequenz gelöst. Die Frequenzanalyse kann ebenfalls in Verbindung mit der Gleichstromanalyse oder alleine benutzt werden.

Das allgemeine Lösungsverfahren, das im Folgenden beschrieben wird, bezieht sich auf Bezeichnungen, die bei allen drei Analysearten vorkommen.

Bei allen drei Methoden muß ein *Netzwerkbaum*[1] gebildet werden, dessen Regeln bei jeder Methode unterschiedlich sind. Durch dieses Verfahren kann man eine B-Matrix aufstellen, die es erlaubt, auf einfache Weise die Spannungen der Verbindungszweige durch Spannungen der Baumzweige und die Ströme durch Baumzweige durch Ströme durch die Verbindungszweige auszudrücken. Tabelle 5.1 zeigt eine zusammengesetzte B-Matrix, die alle möglichen Elementklassen und Matrizen enthält.

Abbildung 5.1: Zusammengesetzte B-Matrix

	Baumzweige				
class	4	5	6	7	Y
1	B_{14}	B_{15}	B_{16}	B_{17}	B_{1Y}
2	B_{24}	B_{25}	B_{26}	B_{27}	B_{2Y}
3	B_{34}	B_{35}	B_{36}	B_{37}	B_{3Y}
8	B_{84}	B_{85}	B_{86}	B_{87}	B_{8Y}
9	B_{94}	B_{95}	B_{96}	B_{97}	B_{9Y}
0	B_{04}	B_{05}	B_{06}	B_{07}	B_{0Y}
X	B_{X4}	B_{X5}	B_{X6}	B_{X7}	B_{XY}

Alle in SCEPTRE vorkommenden Netzwerkelemente werden in Klassen eingeteilt (siehe Tabelle 5.1).

Aus Tabelle 5.1 lassen sich folgende Vektoren und Matrizen herleiten:

R_{22} Diagonalmatrix, bestehend nur aus den Widerständen in den Verbindungszweigen

$G_{22} = R_{22}^{-1}$

R_{55} Diagonalmatrix, bestehend nur aus den Widerständen in den Baumzweigen

$G_{55} = R_{55}^{-1}$

C_{11} Diagonalmatrix, bestehend nur aus den Kapazitäten in den Verbindungszweigen

$S_{11} = C_{11}^{-1}$

C_{44} Diagonalmatrix, bestehend nur aus den Kapazitäten in den Baumzweigen

$S_{44} = C_{44}^{-1}$

L_{33} Matrix, die nur die Induktivitäten und Gegeninduktivitäten in und zwischen den Verbindungszweigen enthält

L_{36} Matrix, die nur die Gegeninduktivitäten zwischen den Baum- und Verbindungszweigen enthält

[1] Als *Baum* bezeichnet man den Graphen, der alle Netzwerkknoten miteinander verbindet, jedoch keine geschlossenen Maschen enthält. Die Verbindungen mit den Netzwerkknoten im Baum heißen *Baumzweige*, alle anderen werden als *Verbindungszweige* bezeichnet. Ein *C-Baum* wird hier definiert als ein Baum, der die Netzwerkelemente in der bevorzugten Reihenfolge E, C, R und L enthält. In dem C-Baum dürfen keine Stromquellen enthalten sein. Diese müssen in den Verbindungszweigen liegen. Als *Schnittmenge* werden die Netzwerkelemente bezeichnet, die das Netzwerk in zwei Knotengruppen teilen, wenn man sie aus dem Netzwerk entfernt.

Tabelle 5.1: Einteilung der Netzwerkelemente in Klassen

	Element
1	Kapazitäten in Verbindungszweigen
2	Widerstände in Verbindungszweigen
3	Induktivitäten in Verbindungszweigen
4	Kapazitäten in Baumzweigen
5	Widerstände in Baumzweigen
6	Induktivitäten in Baumzweigen
7	Spannungsquellen
8	unabhängige Stromquellen
9	Stromquellen 1. Ordnung (spannungsabhängig)[a]
0	Stromquellen 2. Ordnung (abhängig von anderen Stromquellen)[b]
Y	Spannungsquellen, die durch Spannungen an Widerständen gesteuert werden[c]
X	Stromquellen, die durch Ströme durch Widerstände gesteuert werden[d]
D	Spannungsquellen, die bei *adjungierten* Netzwerken mit Stromquellen 2. Ordnung auftreten

[a] treten nur bei der Berechnung der Anfangsbedingungen und bei der AC-Analyse auf
[b] treten nur bei der Berechnung der Anfangsbedingungen auf
[c] werden bei der Zeit- und Frequenzanalyse benötigt
[d] werden bei der TR- und AC-Analyse benötigt.

L_{66} Matrix, die nur die Induktivitäten in den Baumzweigen und die Gegeninduktivitäten zwischen den Baumzweigen enthält

G_{99} Diagonalmatrix, die nur aus den Ableitungen der Spannungen von Stromquellen 1. Ordnung bestehen

I_1, V_1 Vektoren, die nur die Ströme und Spannungen zu Kapazitäten in den Verbindungszweigen enthalten

I_2, V_2 Vektoren, die nur die Ströme und Spannungen zu den Widerständen in den Verbindungszweigen enthalten

I_3, V_3 Vektoren, die nur die Ströme und Spannungen zu den Induktivitäten in den Verbindungszweigen enthalten

I_4, V_4 Vektoren, die nur die Ströme und Spannungen zu Kapazitäten in den Baumzweigen enthalten

I_5, V_5 Vektoren, die nur die Ströme und Spannungen zu Widerständen in den Baumzweigen enthalten

I_6, V_6 Vektoren, die nur die Ströme und Spannungen zu Induktivitäten in den Baumzweigen enthalten

I_8, V_8 Vektoren, die nur die Ströme und Spannungen zu den unabhängigen Stromquellen enthalten

I_9, V_9 Vektoren, die nur die Ströme und Spannungen zu den Stromquellen 1. Ordnung enthalten

I_0, V_0 Vektoren, die nur die Ströme und Spannungen zu den Stromquellen 2. Ordnung enthalten

E_7 Vektor, der nur die Spannungsquellen enthält

5.3 Transiente Lösung

5.3.1 Matrix Operationen

Ein zeitabhängiges System kann man mit Hilfe von *Zustandsvariablen* beschreiben. Es kann gezeigt werden, dass mit den bekannten Kapazitätsspannungen in den Baumzweigen V_4 und den Strömen durch die Induktivitäten in den Verbindungszweigen I_3 alle anderen Ströme und Spannungen des Netzwerkes bestimmt werden können. Mit V_4 und I_3 lässt sich das vollständige Netzwerk durch ein Differentialgleichungssystem 1. Ordnung beschreiben. Ausgehend von den Vektoren V_4 und I_3 lassen sich alle unbekannten Größen $V_1, V_2, V_3, V_5, V_6, I_1, I_2, I_4, I_5, I_6, I_7$ und V_8 ermitteln. Zusätzlich müssen die Ableitungen der Zustandsvariablen für die numerische Integration vorhanden sein, die daraus die neuen Zustandsvariablen für den nächsten Zeitschritt berechnet.

Einige der Gleichungen, die zur Lösung der Unbekannten notwendig sind, können direkt aus der B-Matrix für die TR-Analyse entsprechend Tabelle (5.2) hergeleitet werden. Die B-Matrix ergibt sich

Tabelle 5.2: B-Matrix für die TR-Analyse

Klasse	4	5	6	7
1	B_{14}	0	0	B_{17}
2	B_{24}	B_{25}	0	B_{27}
3	B_{34}	B_{35}	B_{36}	B_{37}
8	B_{84}	B_{85}	B_{86}	B_{87}

aus dem C-Baum, der nur die Elemente E, C, R und L enthält. Man beachte, dass sich die B-Matrix nach Tabelle (5.2) von der B-Matrix nach Tabelle (5.1) darin unterscheidet, dass einige Untermatrizen Null sind[2] und dass keine unterschiedlichen Quellen existieren.

Da sich die Spannungen der Verbindungszweige direkt durch Spannungen der Baumzweige mit Hilfe der B-Matrix ausdrücken lassen, ergeben sich folgende Gleichungen:

$$V_1 = -B_{14}V_4 - B_{17}E_7 \tag{5.1}$$
$$V_2 = -B_{24}V_4 - B_{25}V_5 - B_{27}E_7 \tag{5.2}$$
$$V_3 = -B_{34}V_4 - B_{35}V_5 - B_{36}V_6 - B_{37}E_7 \tag{5.3}$$
$$V_8 = -B_{84}V_4 - B_{85}V_5 - B_{86}V_6 - B_{87}E_7\ . \tag{5.4}$$

Ebenso können die Ströme in den Baumzweigen durch Ströme in den Verbindungszweigen ausgedrückt werden:

$$I_4 = B_{14}^T I_1 + B_{24}^T I_2 + B_{34}^T I_3 + B_{84}^T J_8 \tag{5.5}$$
$$I_5 = B_{25}^T I_2 + B_{35}^T I_3 + B_{85}^T J_8 \tag{5.6}$$
$$I_6 = B_{36}^T I_3 + B_{86}^T J_8 \tag{5.7}$$
$$I_7 = B_{17}^T I_1 + B_{27}^T I_2 + B_{37}^T I_3 + B_{87}^T J_8\ . \tag{5.8}$$

Das ^{T} kennzeichnet eine transponierte Matrix. Aus den Gleichungen (5.1) und (5.7) ergibt sich ebenfalls

$$\dot{V}_1 = -B_{14}\dot{V}_4 - B_{17}\dot{E}_7 \tag{5.9}$$
$$\dot{I}_6 = B_{36}^T \dot{I}_3 + B_{86}^T \dot{J}_8\ . \tag{5.10}$$

[2] Es muss z. B. $B_{15} = 0$ sein, da die Eigenschaften des C-Baumes es verbieten, dass eine Kapazität in einem Verbindungszweig eine Masche mit einem Widerstand in einem Baumzweig bildet.

Man beachte, dass in den letzten beiden Gleichungen die Ableitungen der Quellen, $\dot{E}_7$ und $\dot{J}_8$, benötigt werden. Diese Gleichungen, zusammen mit einigen einfachen Beziehungen, genügen, um alle Netzwerkspannungen und -ströme aus den bekannten Größen herzuleiten.

Berechnung widerstandsbezogener Größen

Dazu gehören die Vektoren I_2, I_5, V_2 und V_5. Aus dem ohmschen Gesetz erhält man

$$V_2 = R_{22} I_2 \tag{5.11}$$
$$V_5 = R_{55} I_5. \tag{5.12}$$

Für I_2 ergibt sich aus den Gleichungen (5.2), (5.6), (5.11) und (5.12)

$$I_2 = M_R^{-1}\{-B_{24}V_4 - B_{27}E_7 - B_{25}R_{55}[B_{35}^T I_3 + B_{85}^T J_8]\} \tag{5.13}$$

mit

$$M_R = R_{22} + B_{25}R_{55}B_{25}^T \; . \tag{5.14}$$

Man erkennt an Gleichung (5.13), dass die Ströme I_2 sich direkt aus den bekannten Zustandsvariablen V_4 und I_3, den Quellen E_7 und J_8 sowie den jeweiligen Untermatrizen der B-Matrix ergeben. Aus I_2 lassen sich dann die Vektoren I_5, V_2 und V_5 mittels der Gleichungen (5.6), (5.11) und (5.12) herleiten.

Alternativ läßt sich V_5 auch aus den Gleichungen (5.2), (5.6), (5.11) und (5.12) bestimmen:

$$V_5 = M_G^{-1}\{-B_{25}^T G_{22}[B_{24}V_4 + B_{27}E_7] + B_{35}^T I_3 + B_{85}^T J_8\} \tag{5.15}$$

mit

$$M_G = G_{55} + B_{25}^T G_{22} B_{25} \; .$$

Den Vektor V_2 erhält man aus Gleichung (5.2), I_2 und I_5 aus

$$I_2 = G_{22}V_2$$
$$I_5 = G_{55}V_5 \; .$$

Die beiden Lösungsansätze unterscheiden sich in der Größe der Matrix, die invertiert werden muss. Bei der ersten Methode muss die Matrix M_R invertiert werden, deren Zeilen- und Spaltenzahlen identisch mit der Anzahl der Klasse-2 Elemente sind, bei der zweiten Methode richten sich die Zeilen- und Spaltenzahlen der Matrix M_G nach der Anzahl der Klasse-5 Elemente. Netzwerke mit konstanten Widerstandswerten benötigen nur eine Matrixinvertierung. In diesem Fall besteht praktisch kein Unterschied zwischen den Methoden. Falls das Netzwerk jedoch mindestens einen variablen Widerstandswert enthält, muss die Matrix für jeden Zeitschritt invertiert werden. Die Größe der Matrix beeinflusst dadurch die benötigte Rechenzeit. SCEPTRE wählt für jedes Netzwerk automatisch den geeigneten Lösungsansatz aus.

Berechnung kapazitätsbezogener Größen

Zu den kapazitätsbezogenen Größen, die zu jedem Zeitschritt berechnet werden müssen, gehören die Vektoren I_1, I_4, V_1 und V_4. Der Vektor V_4 wird durch die Integrationsroutine berechnet. Prinzipiell gelten folgende Beziehungen:

$$\dot{V}_4 = S_{44} I_4 \tag{5.16}$$
$$\dot{V}_1 = S_{11} I_1 \; . \tag{5.17}$$

Aus den Gleichungen (5.5), (5.9), (5.16) und (5.2) erhält man

$$I_1 = M_S^{-1}\{-B_{14} S_{44}[B_{24}^T I_2 + B_{34}^T I_3 + b_{84}^T J_8] - B_{17}\dot{E}_7\} \tag{5.18}$$

mit

$$M_S = S_{11} + B_{14} S_{44} B_{14}^T \; . \tag{5.19}$$

Damit sind alle Größen zur Berechnugn von I_1 bekannt, die Vektoren I_4, V_1 und V_4 ergeben sich entsprechend aus den Gleichungen (5.5), (5.1) und (5.16).

Berechnung induktivitätsbezogener Größen

Dazu gehören die Größen $\dot{I}_3, V_3, V_6$ und I_6. Grundsätzlich gilt

$$V_3 = L_{33}\dot{I}_3 + L_{36}\dot{I}_6 \tag{5.20}$$
$$V_6 = L_{63}\dot{I}_3 + L_{66}\dot{I}_6. \tag{5.21}$$

Kombiniert man die Gleichungen (5.3), (5.10), (5.20) und (5.21) erhält man

$$\dot{I}_3 = M_L^{-1}\{-B_{34} V_4 - B_{35} V_5 - B_{37} E_7 - [B_{36} L_{66} B_{86}^T + L_{36} B_{86}^T]\dot{J}_8\} \tag{5.22}$$

mit

$$M_L = L_{33} + L_{36} B_{36}^T + B_{36} L_{63} + B_{36} L_{66} B_{36}^T \; . \tag{5.23}$$

Aus den Gleichungen (5.10) und (5.20), (5.10) und (5.21) und (5.7) lassen sich die entsprechenden Vektoren V_3, V_6 und I_6 herleiten.

Berechnung der Spannungsquellenströme und Stromquellenspannungen

Diese lassen sich direkt aus den Gleichungen (5.8) und (5.4) bestimmen, da die rechten Seiten zu diesem Zeitpunkt bereits bekannt sind.

5.3.2 Lösungsschema

Die in dem vorangegangenen Kapitel beschriebenen Matrizenoperationen sind geeignet, die Lösung des Zeitverhaltens eines Netzwerkes zu liefern. Dazu müssen diese Operationen zu jedem Integrationsschritt ausgeführt werden. Unter bestimmten Umständen kann der Rechenaufwand vermindert werden, wenn man bestimmte Eigenschaften der B-Matrix ausnutzt.

Fall $B_{14}, B_{25}, B_{36} = 0$

Verschiedene Netzwerkspannungen und -ströme können direkt aus der B-Matrix hergeleitet werden. Die Spannungen an den Verbindungszweigen V_L erhält man aus

$$V_L = -B\, V_{TB} \qquad (V_{TB} = \text{Spannung an den Baumzweigen}) \tag{5.24}$$

die Ströme durch die Baumzweige I_{TB} ergeben sich aus

$$I_{TB} = B\, I_L \qquad (I_L = \text{Ströme durch die Verbindungszweige})\ . \tag{5.25}$$

Wenn die B-Matrix und Vektoren nach Tabelle (5.2) angeordnet sind, führen die Gleichungen (5.24) und (5.25) zu den Gleichungen (5.1) – (5.8). Ist die Untermatrix $B_{25} = 0$, können die Vektoren V_2 und I_5 direkt mit Hilfe der beiden Gleichungen (5.2) und (5.6) bestimmt werden. Sind die widerstandsbezogenen Größen bekannt, können — falls $B_{14} = 0$ ist — die Ströme I_4 mit Gleichung (5.5) berechnet werden. Das gleiche gilt für die Spannungen V_3 mittels der Gleichung (5.3), wenn die Untermatrix $B_{36} = 0$ ist. Falls also die Bedingungen $B_{14}, B_{25}, B_{36} = 0$ erfüllt sind, ergeben sich die Größen V_2, I_5, I_4, V_3, I_6 und V_1 direkt durch Einsetzen ohne Matrizenoperationen. Mit den Gleichungen (5.11), (5.12), (5.18), (5.21), (5.16) und (5.22) erhält man dann die restlichen Größen I_2, V_5, I_1, V_6, V_4 und I_3. Nach diesem Schema generiert SCEPTRE die FORTRAN-Anweisungen, die zu jedem Zeitschritt ohne zusätzliche Matrizenoperationen ausgeführt werden.

Fall $B_{25} \neq 0$

In großeren Netzwerken ist im allgemeinen die Untermatrix $B_{25} \neq 0$. Dann lassen sich I_5 und V_2 nicht durch einfaches Einsetzen berechnen, sondern es müssen die Gleichungen (5.13) oder (5.15) herangezogen werden. Beide Gleichungen beinhalten eine Matrixinvertierung. Die Prozedur läßt sich am besten an einem einfachen Beispiel erklären, zu dem die in Abbildung (5.2) dargestellte B-Matrix gehören soll. In diesem Beispiel sind die Matrizen B_{14} und $B_{36} = 0$, aber $B_{25} \neq 0$ (hervorgehoben in

Abbildung 5.2: Beispiel einer B-Matrix mit $B_{14}, B_{36} = 0, B_{25} \neq 0$

		Klasse 4		Klasse 5				Klasse 7
		C_1	C_2	R_A	R_B	R_C	R_D	E_1
Klasse 2	R_1	1	0	**1**	**0**	**0**	**0**	-1
	R_2	0	1	**0**	**1**	**1**	**0**	0
	R_3	0	1	**0**	**0**	**0**	**0**	-1
	R_4	0	0	**1**	**0**	**0**	**0**	0
Klasse 3	L_1	0	0	0	0	0	1	0
	L_2	0	0	1	0	0	0	0

Abbildung (5.2)). Die Widerstände R_3 und R_D leisten keinen Beitrag in B_{25}, und durch Einsetzen erhält man direkt

$$V_{R3} = E_1 - V_{C2}$$
$$I_{RD} = I_{L1}\ .$$

Folglich sind

$$I_{R3} = (E_1 - V_{C2})/R_3$$
$$V_{RD} = I_{L1} RD \; .$$

Diese Größen werden zu jedem Zeitschritt ohne Matrixinversion berechnet. Die anderen widerstandsbezogenen Größen werden in der herkömmlichen Weise entsprechend der Gleichungen (5.13) und (5.14) ermittelt. Anschließend werden die verbleibenden Netzwerkgrößen bestimmt. Die einzigen Matrixinversionen, die zu jedem Integrationsschritt wiederholt werden müssen, sind die zur Berechnung der Ströme I_{R1}, I_{R2} und I_{R4}.

5.3.3 Zeitliche Ableitungen von Quellen

In den Gleichungen (5.9) und (5.10) werden die zeitlichen Ableitungen der Quellen $\dot{E}_7$ und $\dot{J}_8$ benötigt. Diese müssen unter bestimmten Bedingungen vom Benutzer vorgegeben werden.

Veränderliche Spannungsquelle und $B_{17} \neq 0$

Falls die Untermatrix $B_{17} \neq 0$ und die Spannungsquelle konstant sind, setzt SCEPTRE die zeitliche Ableitung auf Null. Wird eine von Null verschiedene Ableitung benötigt und existiert sie nicht, bricht der Lauf mit einer entsprechenden Fehlermeldung ab. Dieser Fall tritt z. B. bei Maschen auf, die nur Kapazitäten und mindestens eine veränderliche Spannungsquelle enthalten.

Veränderliche Stromquelle und $B_{86} \neq 0$

Das gleiche gilt sinngemäß für Stromquellen und die Untermatrix B_{86}. Dieses Problem tritt bei Schnittmengen auf, die nur aus Induktivitäten und mindestens einer variablen Stromquelle bestehen.

5.3.4 Linear abhängige Quellen

SCEPTRE bietet die Möglichkeit, linear abhängige Quellen zu simulieren. Damit kann der Benutzer Spannungs- und Stromquellen definieren, die linear von Widerstandsspannungen und -strömen abhängig sind. Ein typisches Anwendungsgebiet sind die Kleinsignal-Transistorersatzschaltungen (siehe Abb. (5.3)).

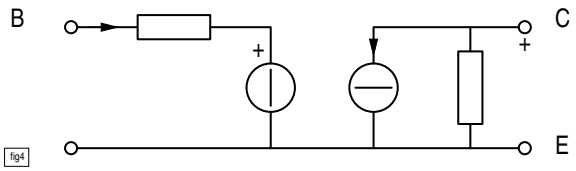


Abbildung 5.3: Kleinsignal-Transistorersatzschaltbild mit h-Parametern

Diese linear abhängigen Quellen bedürfen einer gesonderten Betrachtung. Da sie nicht direkt von den Zustandsvariablen abgeleitet werden können und somit zu Beginn eines Integrationsschrittes nicht bekannt sind, müßten sie normalerweise auf die Werten des vorherigen Integrationsschrittes zugreifen, was zu einem sogenannten *„computational delay“* führen würde (siehe Kapitel (3.7)).

In der B-Matrix wird dies durch die Klassen Y und X berücksichtigt (siehe Tabelle (5.1) und (5.4)).

Abbildung 5.4: B-Matrix mit linear abhängigen Quellen

Klasse	4	5	6	7	Y
1	B_{14}	0	0	B_{17}	B_{1Y}
2	B_{24}	B_{25}	0	B_{27}	B_{2Y}
3	B_{34}	B_{35}	B_{36}	B_{37}	B_{3Y}
8	B_{84}	B_{85}	B_{86}	B_{87}	B_{8Y}
X	B_{X4}	B_{X5}	B_{X6}	B_{X7}	B_{XY}

Damit erweitern sich die Gleichungen (5.2) und (5.6) zu

$$V_2 = -B_{24}V_4 - B_{25}V_5 - B_{27}E_7 - B_{2Y}E_Y \tag{5.26}$$
$$I_5 = B_{25}^T I_2 + B_{35}^T I_3 + B_{85}^T J_8 + B_{X5}^T J_X \, . \tag{5.27}$$

Da jede Spannungsquelle der Klasse Y von einer Spannung an einem Widerstand im Netzwerkbaum oder in den Verbindungszweigen abhängt, gilt

$$E_Y = k_1 V_2 + k_2 V_5 \tag{5.28}$$

mit

k_1 Matrix mit konstanten Elementen, enthält (Anzahl der Spannungsquellen Klasse Y) Reihen und (Anzahl der Elemente Klasse 2) Spalten

k_2 Matrix mit konstanten Elementen, enthält (Anzahl der Spannungsquellen Klasse Y) Reihen und (Anzahl der Elemente Klasse 5) Spalten

E_Y Vektor mit den Spannungsquellen Klasse Y.

Analog dazu hängt jede Stromquelle der Klasse X von einem Strom durch einen Widerstand im Netzwerkbaum oder in den Verbindungszweigen ab:

$$J_X = k_3 I_2 + k_4 I_5 \tag{5.29}$$

mit

k_3 Matrix mit konstanten Elementen, enthält (Anzahl der Stromquellen Klasse X) Reihen und (Anzahl der Elemente Klasse 2) Spalten

k_4 Matrix mit konstanten Elementen, enthält (Anzahl der Stromquellen Klasse X) Reihen und (Anzahl der Elemente Klasse 5) Spalten

J_X Vektor mit den Stromquellen Klasse X.

Werden die Gleichungen (5.28), (5.29), (5.11) und (5.12) in die Gleichungen (5.26) und (5.27) eingesetzt, erhält man

$$R_{22}I_2 + B_{25}V_5 + B_{2Y}[k_1 R_{22} I_2 + k_2 V_5] = -B_{24}V_4 - B_{27}E_7 \tag{5.30}$$
$$G_{55}V_5 - B_{25}^T I_2 - B_{X5}^T[k_3 I_2 + k_4 G_{55} V_5] = B_{35}^T I_3 + B_{85}^T J_8 \tag{5.31}$$

oder

$$\begin{bmatrix} R_{22} + B_{2Y}k_1R_{22} & B_{25} + B_{2Y}k_2 \\ -B_{25}^T - B_{X5}^T k_3 & G_{55} - B_{X5}^T k_4 G_{55} \end{bmatrix} \begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = \begin{bmatrix} -B_{24}V_4 - B_{27}E_7 \\ B_{35}^T I_3 + B_{85}^T J_8 \end{bmatrix}.$$

Kürzt man die Matrix auf der linken Seite mit M_{RG} ab, ergibt sich für die verzögerungsfreie Berechnung[3] von I_2 und V_5

$$\begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = M_{\mathrm{RG}}^{-1} \begin{bmatrix} -B_{24}V_4 - B_{27}E_7 \\ B_{35}^T I_3 + B_{85}^T J_8 \end{bmatrix}.$$

Da die Matrizen k_1 – k_4 nur konstante Elemente enthalten, braucht die Matrix M_{RG} nur einmal invertiert zu werden. Sind die Widerstände nicht konstant, muss M_{RG} natürlich für jeden Integrationsschritt invertiert werden.

Durch die zusätzlichen Klassen X und Y für die linear abhängigen Quellen entsprechend Tabelle (5.4) müssen die Gleichungen (5.5), (5.3), (5.4) und (5.8) erweitert werden:

$$I_4 = B_{14}^T I_1 + B_{24}^T I_2 + B_{34}^T I_3 + B_{84}^T J_8 + B_{X4}^T J_X \tag{5.32}$$
$$V_3 = -B_{34}V_4 - B_{35}V_5 - B_{36}V_6 - B_{37}E_7 - B_{3Y}E_Y \tag{5.33}$$
$$V_8 = -B_{84}V_4 - B_{85}V_5 - B_{86}V_6 - B_{87}E_7 - B_{8Y}E_Y \tag{5.34}$$
$$I_7 = B_{17}^T I_1 + B_{27}^T I_2 + B_{37}^T I_3 + B_{87}^T J_8 + B_{X7}^T J_X. \tag{5.35}$$

Zusätzlich erhält man noch die beiden Beziehungen

$$V_X = -B_{X4}V_4 - B_{X5}V_5 - B_{X7}E_7 - B_{XY}E_Y \tag{5.36}$$
$$I_Y = B_{2Y}^T I_2 + B_{3Y}^T I_3 + B_{8Y}^T J_8 + B_{XY}^T J_X\ . \tag{5.37}$$

Es gibt jedoch eine Einschränkung hinsichtlich der Benutzung von linear abhängigen Quellen: die erweiterte B-Matrix nach Tabelle (5.4) ergänzt die beiden Gleichungen (5.9) und (5.10) zu

$$\dot{V}_1 = -B14\dot{V}_4 - R_{17}\dot{E}_7 - B_{1Y}\dot{E}_Y \tag{5.38}$$
$$\dot{I}_6 = B_{36}^T \dot{I}_3 + B_{86}^T \dot{J}_8 + B_{X6}^T \dot{J}_X\ . \tag{5.39}$$

Man erkennt, dass die zeitlichen Ableitungen $\dot{E}_Y$ und $\dot{J}_X$ vorhanden sein müssen, wenn B_{1Y} oder $B_{X6} \neq 0$ ist. Differenziert man die Gleichungen (5.28) und (5.29), werden die Ableitungen $\dot{V}_2, \dot{V}_5, \dot{I}_2$ und $\dot{I}_5$ benötigt. Die sind SCEPTRE aber nicht bekannt! Deshalb prüft SCEPTRE, ob B_{1Y} oder $B_{X6} \neq 0$ ist und bricht mit einer entsprechenden Fehlermeldung ab. Die Untermatrix B_{1Y} ist dann ungleich Null, wenn in einer Masche nur Kapazitäten und mindestens eine gesteuerte Spannungsquelle vorhanden sind. Gibt es eine Schnittmenge, die nur aus Induktivitäten und mindestens einer gesteuerten Stromquelle besteht, gilt analog $B_{X6} \neq 0$.

5.3.5 Integrationsverfahren

Zum Lösen des Zustandsgleichungssystems bietet SCEPTRE wahlweise drei explizite (`XPO,RUK,TRAP`) und ein implizites (`IMPLICIT`) Integrationsverfahren an. Das Runge-Kutta- und das Trapezverfahren wurden mit Modifikationen von anderen Netzwerkprogrammen übernommen. Das Exponentialverfahren wurde am IBM Scientific Center in Palo Alto, Kalifornien, entwickelt und an SCEPTRE angepasst. Es hat sich gezeigt, dass dieses Verfahren fast immer schneller ist, als die anderen. Wird unter `RUN CONTROLS` kein Verfahren angegeben, wird automatisch `XPO` gewählt. Das implizite Verfahren bewährt sich am besten, wenn in einem Netzwerk gleichzeitig sehr große und sehr kleine Zeitkonstanten auftreten.

[3] „computational delay“

Das Runge-Kutta-Verfahren `RUK`

Für das Differentialgleichungssystem

$$\dot{y} = f(t, y)$$

liefert das Runge-Kutta-Verfahren vierter Ordnung die numerische Lösung

$$y(t+h) = y(t) + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

mit

$$\begin{aligned} k_1 &= hf[t, y(t)] \\ k_2 &= hf\left[t + \frac{h}{2}, y(t) + \frac{1}{2}k_1\right] \\ k_3 &= hf\left[t + \frac{h}{2}, y(t) + \frac{1}{2}k_2\right] \\ k_4 &= hf[t + h, y(t) + k_3]\ . \end{aligned}$$

Die variable Schrittweite wird mit

$$k_5 = hf[t + h, y(t + h)]$$

und

$$E = k_1 - 2k_3 - 2k_4 + 3k_5$$

gesteuert. Die o. a. Beziehungen lassen sich in ein Differentialgleichungssystem überführen der Form

$$\begin{aligned} y(t) &= [y_1(t), \cdots, y_n(t)] \\ E &= (e_1, \cdots, e_n)\ . \end{aligned}$$

Mit den Fehlerschranken

$$u_1, u_2, l_1, l_2 \geq 0$$

bestimmt man

$$\begin{aligned} U_k &= u_1 + u_2\,|y_k(t+h)| \\ L_k &= l_1 + l_2\,|y_k(t+h)| \end{aligned}$$

für $k = 1, \ldots, n$. Die Schrittweite errechnet sich folgendermaßen:

- gilt $|e_k| > 1.5U_k$ für beliebiges k, dann wird die Schrittweite h halbiert, die unabhängige Variable wird von $t + h$ auf t, und die Werte von y und $\dot{y}$ auf ihre Werte zur Zeit t gesetzt;
- gilt $|e_k| > 0.75U_k$ für beliebiges k und $|e_k| \leq 1.5U_k$ für alle k, dann bleibt die Schrittweite für diesen Integrationsschritt, wird aber für den nächsten Schritt halbiert;
- gilt $L_k \leq |e_k| \leq 0.75U_k$ für alle k, bleibt die Schrittweite unverändert;
- gilt $|e_k| \leq U_k$ für alle k und $|e_k| < L_k$ für irgendein k, dann wird die Schrittweite nach sieben Integrationsschritten verdoppelt.

Nach einer Halbierung oder einer Verdoppelung müssen mindestens sieben Integrationsschritte vergehen, bis die Schrittweite wieder verdoppelt werden kann . Für die absoluten Fehlerschranken empfehlen sich die Werte[4]

$$u_1 = 0.0075,\ u_2 = 0,\ l_1 = 0.005,\ l_2 = 0,$$

und für die relativen Fehlerschranken

$$u_1 = 0.005,\ u_2 = 0.005,\ l_1 = 0.005,\ l_2 = 0.005.$$

Das modifizierte Trapezverfahren `TRAP`

Methode Ausgehend von der Differentialgleichung

$$\dot{y}(t) = f(t, y(t)), y(0) = y_0$$

ergibt sich folgendes Integrationsverfahren:

$$y_p\left(t + \frac{3}{2}h\right) = y(t) + \frac{h}{2}\dot{y}(t) \tag{5.40}$$

$$\dot{y}_p\left(t + \frac{3}{2}h\right) = f\left[t + \frac{3}{2}h, y_p(t + \frac{3}{2}h)\right] = f\left[t + \frac{3}{2}h, y(t) + \frac{h}{2}\dot{y}(t)\right] \tag{5.41}$$

$$y_c(t + h) = y(t) + \frac{h}{3}\left[2\dot{y}(t) + \dot{y}_p\left(t + \frac{3}{2}h\right)\right] \tag{5.42}$$

$$\dot{y}_c(t + h) = f[t + h, y_c(t + h)]\ . \tag{5.43}$$

Die Integrationsschrittweite wird aus $|E|$ abgeleitet mit

$$E = \frac{2}{3}h\left|\dot{y}_p(t + \frac{3}{2}h) - \dot{y}(t)\right|\ . \tag{5.44}$$

Rundungsfehler Wenn gilt

$$\ddot{y} = f_t + \dot{y}f_y$$

und die wahre Lösung y_T angenähert wird durch

$$y_T(t + h) = y(t) + h\dot{y}(t) + \frac{h^2}{2}\ddot{y}(t)$$

erhält man

$$y_T(t + h) = y(t) + h\dot{y}(t) + \frac{h^2}{2}(f_T + \dot{y}f_y)\ . \tag{5.45}$$

Außerdem gilt

$$f\left[t + \frac{3h}{2}, y(t) + \frac{h}{2}\dot{y}(t)\right] \doteq f(t, y(t)) + \frac{3h}{2}f_t + \frac{h}{2}\dot{y}f_y\ , \tag{5.46}$$

so dass für Gleichung (5.42) geschrieben werden kann

$$y_c(t + h) \doteq y(t) = \frac{h}{3}\left[2\dot{y}(t) + \dot{y}(t) + \frac{3h}{2}f_t + \frac{h}{2}\dot{y}f_y\right]\ .$$

[4] können mit den Anweisungen `MAXIMUM/MINIMUM ABSOLUTE/RELATIVE ERROR` unter `RUN CONTROLS` gesetzt werden

Subtrahiert man Gleichung (5.42) von (5.45) ergibt sich für den ungefähren lokalen Rundungsfehler

$$y_T(t+h) - y_c(t+h) \doteq \left(\frac{h^2}{2} - \frac{h^2}{6}\right) \dot{y} f_y = \frac{h^2}{3} \dot{y} f_y \; .$$

Aus den Gleichungen (5.41) und (5.46) erhält man

$$\dot{y}_p \left(t + \frac{3h}{2}\right) - \dot{y}(t) \doteq \frac{3h}{2} f_t + \frac{h}{2} \dot{y} f_y$$

und somit

$$\frac{2h}{3} \left[\dot{y}(t + \frac{3h}{2}) - \dot{y}(t)\right] \doteq h^2 f_t + \frac{h^2}{3} \dot{y} f_y \; . \tag{5.47}$$

Wenn $f_t = 0$ ist, kann die Gleichung (5.47) zur Abschätzung des lokalen Rundungsfehlers bei Vernachlässigung von Termen höherer Ordnung herangezogen werden. Erfahrungen haben gezeigt, dass bei Differentialgleichungssystemen der Art $\dot{y} = A(y)y + B$, bei dem A stückweise linear ist, diese Methode und Schrittweitensteuerung ausreichend sind.

Stabilität Gegeben sei ein Differentialgleichungssystem $\dot{y} = -ay$ mit $y(0) = y_0$ und $a > 0$. Aus dem Integrationsschema nach den Gleichungen (5.40) – (5.43) ergibt sich

$$y_c(nh) = \left(1 - ah + \frac{(ah)^2}{6}\right)^n y(0) \; .$$

Dies kann durch Induktion nachgewiesen werden. Somit gilt für die numerische Stabilität die notwendige und hinreichende Bedingung

$$\left|1 - ah + \frac{(ah)^2}{6}\right| < 1, \; 0 < ah < 6 \; .$$

Die Bedingung $|ah| < 6$ ist notwendig und hinreichend für die numerische Stabilität (Abb. 5.5), d. h. das modifizierte Trapezverfahren besitzt den Stabilitätsradius $r = 6$. Weiterhin erkennt man aus den

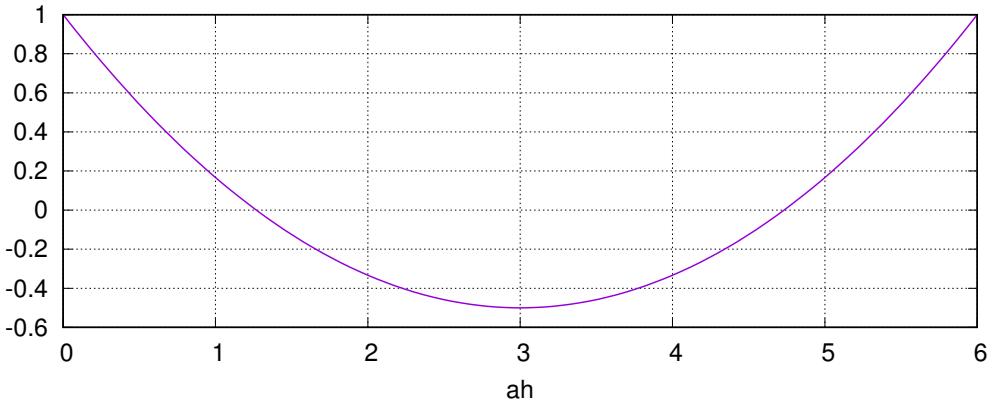


Abbildung 5.5: Bedingung $0 < ah < 6$ für die numerische Stabilität des TRAP-Verfahrens

Gleichungen (5.40) und (5.41), dass pro Integrationsschritt zwei Ableitungen gebildet werden müssen, das Verfahren hat also die Schrittzahl $p = 2$. Das Verhältnis r/p ist ein Maß für die Effizienz der Integrationsmethode und beschreibt die minimale Anzahl von Integrationsschritte pro Zeitinterval. Je höher das Verhältnis r/p ist, desto weniger Integrationsschritte sind für ein vorgegebenes a erforderlich.

Die Tabelle 5.3 zeigt typische Werte für r und p für verschiedene Integrationsverfahren.

Tabelle 5.3: Das Verhältnis r/p für verschiedene Integrationsverfahren

Verfahren	r	p	r/p
modifiziertes TRAP	6	2	3
Euler [7]	2	1	2
Trapez [8]	2	2	1
NIDE (1-Schritt) [9]	0.8	1	0.8
Runge-Kutta 4. Ordnung	2.78	4	0.7
NIDE (2-Schritt)	1	2	0.5
Hamming [10]	0.85	2	0.425

Schrittweitensteuerung Wie bereits erwähnt, wird die Größe E_2 (siehe Gleichung (5.44)) zur Schrittweitensteuerung herangezogen. Im Prinzip funktioniert diese Steuerung wie beim Runke-Kutta-Verfahren. Mit den Fehlerschranken

$$u_1, u_2, l_1, l_2 \geq 0$$

berechnet man

$$U = u_1 + u_2 \left|y_c(t+h)\right| \ , \qquad L = l_1 + l_2 \left|y_c(t+h)\right| \ .$$

- Falls $|E_2| > 0.75U$, wird die Schrittweite verringert.
- Falls $|E_2| < L$, wird die Schrittweite vergrößert.
- Falls $L \leq |E_2| \leq 0.75U$, wird die Schrittweite nicht verändert.

Da E_2 annähernd dem lokalen Rundungsfehler entspricht, empfehlen sich für die absoluten Fehlerschranken folgende Werte:

$$u_1 = 10^{-2},\ l_1 = 5 \cdot 10^{-4},\ u_2 = 0,\ l_2 = 0.$$

Für die relative Fehlersteuerung haben sich ebenfalls folgende Werte bewährt:

$$u_1 = 10^{-3},\ l_1 = 5 \cdot 10^{-5},\ u_2 = 10^{-2},\ l_2 = 5 \cdot 10^{-4}.$$

Zur Halbierung der Schrittweite müssen die o. a. Werte um den Faktor 4 verkleinert werden.

Das Exponentialverfahren `XPO`

Die Methode Ausgehend von der Differentialgleichung

$$\dot{y}(t) = f[t, y(t)],\ y(0) = y_0$$

liefert dieses Integrationsverfahren die Lösung

$$y(t+h) = y(t) + \dot{y}_A(t)h + \frac{e^{\lambda h} - 1}{\lambda h} h \dot{y}_p(t) \tag{5.48}$$

außer in zwei Fällen, die jedoch später behandelt werden. In Gleichung (5.48) bedeuten

$$\dot{y}_A(t) = \frac{y(t) - y(t - h_0)}{h_0}$$

mit $t - h_0$ als letzten Integrationszeitpunkt, sowie

$$\dot{y}_p(t) = \dot{y}(t) - \dot{y}_A(t) \tag{5.49}$$

$$\lambda = \frac{\ddot{y}_p(t)}{\dot{y}_p(t)} \,. \tag{5.50}$$

Da die zweite Ableitung $\ddot{y}(t)$ nicht explizit durch das Programm berechnet werden kann, approximiert man $\ddot{y}(t)$ durch $\ddot{y}_\delta(t)$:

$$\begin{aligned} y_\delta(t+\delta) &= y(t) + \delta\dot{y}(t) \\ \dot{y}_\delta(t+\delta) &= f[t+\delta, y_\delta(t+\delta)] \\ \ddot{y}_\delta(t) &= \frac{\dot{y}_\delta(t+\delta) - \dot{y}(t)}{\delta} \end{aligned} \tag{5.51}$$

mit $0 < \delta \leq \frac{h}{4}$. Die erste Ausnahme tritt ein, wenn $\ddot{y}_\delta(t)$ und $\dot{y}_p(t)$ gleiches Vorzeichen besitzen. Setzt man $\dot{y}_A(t) = 0$, erhält man aus den Gleichungen (5.49) und (5.50)

$$\dot{y}_p(t) = \dot{y}(t) \tag{5.52}$$

$$\lambda = \frac{\ddot{y}_\delta(t)}{\dot{y}(t)} \,. \tag{5.53}$$

Damit erhält man eine bessere Annäherung zur Lösung von $\ddot{y} = \lambda y + b$, ohne die generelle Qualität dieser Integrationsmethode zu verschlechtern.

Die zweite Ausnahme tritt auf, wenn λ — entweder aus Gleichung (5.50) oder (5.52) berechnet — nicht negativ wird. In diesem Fall wird der Ausdruck $\frac{e^{\lambda h}-1}{\lambda h}$ in Gleichung (5.48) durch den Term $1 + \frac{\lambda h}{2}$ ersetzt und es ergibt sich

$$y(t+h) = y(t) + \dot{y}_A(t)h + \left(1 + \frac{\lambda h}{2}\right) h\dot{y}_p(t) \,. \tag{5.54}$$

Rundungsfehler Entwickelt man den Ausdruck $\frac{e^x-1}{x}$ in eine Taylor-Reihe, erhält man

$$\frac{e^x - 1}{x} = 1 + \frac{x}{2} + \frac{x^2}{6} + O(x^3)$$

Gleichung (5.48) lautet dann

$$\begin{aligned} y(t+h) &= y(t) + \dot{y}_A(t)h + \dot{y}_p(t)h + \frac{h^2}{2}\lambda\dot{y}_p(t) + \frac{h^3\lambda^2\dot{y}_p(t)}{6} + O(h^4) \\ &= y(t) + \dot{y}(t)h + \frac{h^2}{2}\ddot{y}_\delta(t) + \frac{h^3}{6}\lambda\ddot{y}_\delta(t) + O(h^4) \end{aligned} \tag{5.55}$$

Schreibt man für Gleichung (5.51)

$$\ddot{y}_\delta(t) = \ddot{y}(t) + \frac{\delta}{2}\left[\frac{d^3y}{dt^3} - \ddot{y}(t)\frac{\partial f}{\partial y}\right] + O(\delta^2) \tag{5.56}$$

und setzt man dies in Gleichung (5.55) ein, erhält man

$$y(t+h) = y(t) + \dot{y}(t)h + \ddot{y}(t)\frac{h^2}{2} + \frac{h^2}{4}\delta\left[\frac{d^3y}{dt^3} - \ddot{y}(t)\frac{\partial f}{\partial y}\right] + O(h^3) \tag{5.57}$$

Da $\delta \leq \frac{h}{4}$, bewegt sich der Fehler in der Größenordnung $O(h^3)$.

Schrittweitensteuerung Es soll $y_T(t+\xi)$ die wahre Lösung der Differentialgleichung darstellen mit $t \leq t+\xi \leq t+h$, weiterhin soll $y_c(t+\xi)$ die berechnete Lösung sein, $\dot{y}_e(t+\xi)$ die Ableitung von $y_c(t+\xi)$ nach ξ, d. h.

$$\dot{y}_e(t+\xi) = \begin{cases} \dot{y}_A(t) + (1+\lambda\xi)\dot{y}_p(t) & \lambda \geq 0 \\ \dot{y}_A(t) + e\lambda\xi\dot{y}_p(t) & \lambda < 0 \end{cases} .$$

Außerdem soll $\dot{y}_c(t+\xi)$ die Ableitung des Ausdrucks $y_c(t+\xi)$ sein.

Im Idealfall läßt sich die Schrittweite aus dem Ausdruck

$$E_0 = \int_0^h [\dot{y}_T(t+\xi) - \dot{y}_e(t-\xi)]\,\mathrm{d}\,\xi$$

berechnen, jedoch ist $\dot{y}_T(t+\xi)$ außer für $\xi = 0$ nicht bekannt. Nutzt man die Tatsache aus, dass

$$\begin{aligned} \dot{y}_c(t+\xi) &= f\{t+\xi, y_T(t+\xi) + [y_c(t+\xi) - y_T(t+\xi)]\} \\ &= \dot{y}_T(t+\xi) + O(\xi^3) \end{aligned} \tag{5.58}$$

erhält man

$$E_0 = \int_0^h [\dot{y}_c(t+\xi) - \dot{y}_e(t+\xi)]\,\mathrm{d}\,\xi + \int_0^h [\dot{y}_T(t+\xi) - \dot{y}_c(t+\xi)]\,\mathrm{d}\,\xi \tag{5.59}$$

$$= \int_0^h [\dot{y}_c(t+\xi) - \dot{y}_e(t+\xi)]\,\mathrm{d}\,\xi + O(h^4) \tag{5.60}$$

Wenn $\ddot{y}_c(t+\xi) - \ddot{y}_e(t+\xi)$ für $0 \leq \xi \leq h$ nicht sein Vorzeichen wechselt, gilt

$$|E_0| \leq h|\dot{y}_T(t+h) - \dot{y}_e(t+h)| \leq h|\dot{y}_c(t+h) - \dot{y}_e(t+h)| + |O(h^4)| \tag{5.61}$$

Somit erhält man für kleine Werte von h den Rundungsfehler zu

$$E = h|\dot{y}_c(t+h) - \dot{y}_e(t+h)|.$$

Mit den Fehlerschranken

$$u_1, u_2, l_1, l_2 \geq 0$$

berechnet man

$$\begin{aligned} U &= u_1 + u_2\,|y_c(t+h)| \\ L &= l_1 + l_2\,|y_c(t+h)| \ . \end{aligned}$$

Für die Schrittweite gilt:

- falls $|E| > U$, wird die Schrittweite verkleinert
- falls $|E| < L$, wird die Schrittweite vergrößert
- falls $L \geq E \geq U$, wird die Schrittweite nicht verändert.

Für die absoluten Fehlerschranken empfehlen sich die Werte

$$u_1 = 75 \cdot 10^{-3},\ l_1 = 2 \cdot 10^{-4},\ u_2 = 0,\ l_2 = 0$$

und als relative Fehlerschranken

$$u_1 = 5 \cdot 10^{-3},\ l_1 = 1 \cdot 10^{-4},\ u_2 = 5 \cdot 10^{-3},\ l_2 = 2 \cdot 10^{-4}.$$

Absolute und relative Fehlerschranken

Alle drei bisher beschriebenen expliziten Integrationsmethoden sind Näherungsverfahren zum Lösen von Differentialgleichungen, die die Zustandsvariablen zu jedem Integrationszeitpunkt bestimmen. Jede Methode verfügt über eine automatische Schrittweitensteuerung. Dabei werden Schrittweite und die Ableitungen der Zustandsvariablen mit einer Kontrollgröße verglichen (siehe z. B. Gleichung (5.44)). Diese sog. Fehlerschranke kann eine Konstante oder eine Funktion der Zustandsvariablen sein. Im ersten Fall spricht man von einer absoluten, andernfalls von einer relativen Fehlerschranke.

Liegen die Zustandsvariablen größenmäßig weit auseinander, ist es nicht sinnvoll, alle Zustandsvariablen mit der gleichen, absoluten Genauigkeit zu berechnen. Bei Vorgabe einer absoluten Genauigkeit wird unnötigerweise eine zu kleine Schrittweite für die wertmäßig größeren Zustandsvariablen verwendet. Benutzt man hingegen eine relative Fehlerschranke, wird dieser Nachteil umgangen.

Allgemein kann man daraus schließen, dass absolute Fehlerschranken nur dann sinnvoll sind, wenn alle Zustandsvariablen kleine Werte besitzen. Streuen ihre Werte, benutzt man vorteilhafter relative Fehlerschranken.

Alle drei expliziten Verfahren benutzen relative Fehlerschranken, aber der Benutzer kann ohne weiteres auch absolute Fehlerschranken angeben (siehe [2, Kapitel `RUN CONTROLS`]). Kontrolliert werden die Fehler durch die Variablen u und l. Setzt man für die oberen Fehlerschranken u_1 oder u_2 große Werte ein, sinkt die Wahrscheinlichkeit, dass die Integrationsschrittweite halbiert wird. Werden die unteren Fehlerschranken l_1 und l_2 vergrößert, steigt die Wahrscheinlichkeit, dass die Integrationsschrittweite vergrößert wird. Grundsätzlich steigt mit der Genauigkeit auch die Rechenzeit.

Das implizite Verfahren `IMPLICIT`

Allgemeine Formulierung Die allgemeine Formulierung aller impliziten Integrationsmethoden lautet:

$$Y(n+1) = \sum_{i=0}^{p} a_i Y(n-i) + h \sum_{i=-1}^{p} b_i \dot{Y}(n-1) \; . \tag{5.62}$$

Dabei stellen Y einen Vektor der Zustandsvariablen, $\dot{Y}$ dessen Ableitung, n die Schrittnummer, h die Schrittweite und a_i und b_i Konstanten dar. Ist $p > 0$, handelt es sich um ein Mehrschrittverfahren. Das sog. implizite Euler-Verfahren stellt eine vereinfachte Form von Gleichung (5.62) dar:

$$Y(n+1) = Y(n) + h\dot{Y}(n+1) \; . \tag{5.63}$$

Soll ein Differentialgleichungssystem $m.$ Ordnung mit diesem Verfahren gelöst werden, erhält man die allgemeine Matrizengleichungen:

$$\begin{bmatrix} 1 - hg\frac{\partial \dot{Y}_1}{\partial Y_1}(Y_1,\dots,Y_m,t) & \dots & -hg\frac{\partial \dot{Y}_1}{\partial Y_m}(Y_1,\dots,Y_m,t) \\ \vdots & & \vdots \\ -hg\frac{\partial \dot{Y}_m}{\partial Y_1}(Y_1,\dots,Y_m,t) & \dots & 1 - hg\frac{\partial \dot{Y}_m}{\partial Y_m}(Y_1,\dots,Y_m,t) \end{bmatrix} \begin{bmatrix} \Delta Y_1^k \\ \vdots \\ \Delta Y_m^k \end{bmatrix} = \begin{bmatrix} -F_l(Y_1,\dots,Y_m,t) \\ \vdots \\ -F_m(Y_1,\dots,Y_m,t) \end{bmatrix} . \tag{5.64}$$

Dieses iterative Verfahren wird solange ausgeführt, bis Konvergenz eintritt. Der Index k stellt die $k.$ Näherung dar, g ist eine von der Ordnung abhängige Konstante und F_i eine Funktion der $i.$ Differentialgleichung, Schrittweite und der m Zustandsvariablen. Bei großen Problemen wird zur Lösung von Gleichung (5.64) die *Sparse-Matrix-Technik* angewendet.

Die Jacobi Matrix Während die zeitlichen Ableitungen der Zustandsvariablen $\dot{Y}_i$ dem Programm bekannt sind, müssen die partiellen Ableitungen

$$\frac{\partial \dot{Y}_i}{\partial Y_j}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq m$$

berechnet werden. Dabei geht man von der allgemeinen Matrix Gleichung

$$\dot{Y} = AY + BU + N\dot{U} \tag{5.65}$$

aus. Wie man sieht, erhält man die gesuchten partiellen Ableitungen aus der Matrix A durch partielle Differenzierung von Gleichung (5.65).

Im Folgenden wird die symbolische Form der allgemeinen A-Matrix hergeleitet. Zunächst setzt man die Gleichungen (5.9) und (5.13) in die Gleichung (5.5) ein und erhält:

$$\begin{aligned} C_{44}\dot{V}_4 = {} & B_{14}^T C_{11} \left(-B_{14}\dot{V}_4 - B_{17}\dot{E}_7\right) \\ & + B_{24}^T M_R^{-1} \left(-B_{24}V_4 - B_{27}E_7 - B_{25}R_{55}B_{35}^T I_3 - B_{25}R_{55}B_{85}^T J_8\right) \\ & + B_{34}^T I_3 + B_{84}^T J_8 \end{aligned} \tag{5.66}$$

und durch weitere Umformung

$$\begin{aligned} (C_{44} + B_{14}^T C_{11} B_{14})\dot{V}_4 = {} & -B_{14}^T C_{11} B_{17}\dot{E}_7 \\ & - B_{24}^T M_R^{-1} \left(B_{24}V_4 + B_{27}E_7 + B_{25}R_{55}B_{35}^T I_3 + B_{25}R_{55}B_{85}^T J_8\right) \\ & + B_{34}^T I_3 + B_{84}^T J_8 \ . \end{aligned} \tag{5.67}$$

Desweiteren werden die Gleichungen (5.21), (5.10), (5.20) und (5.22) in Gleichung (5.3) eingesetzt und es ergibt sich:

$$\begin{aligned} L_{33}\dot{I}_3 + L_{36}\left(B_{36}^T\dot{I}_3 + B_{86}^T\dot{J}_8\right) = {} & \\ & - B_{34}V_4 - B_{35}M_G^{-1}\left\{-B_{25}^T G_{22}(B_{24}V_4 + B_{27}E_7) + B_{35}^T I_3 + B_{85}^T J_8\right\} \\ & - B_{36}(L_{63} + L_{66}B_{36}^T)\dot{I}_3 - B_{36}L_{66}B_{86}^T\dot{J}_8 - B_{37}E_7 \end{aligned} \tag{5.68}$$

und durch weitere Umformung

$$\begin{aligned} \left[L_{33} + L_{36}B_{36}^T + B_{36}L_{63} + B_{36}L_{66}B_{36}^T\right]\dot{I}_3 = {} & -B_{34}V_4 \\ & - B_{35}M_G^{-1}\left(-B_{25}^T G_{22}B_{24}V_4 - B_{25}^T G_{22}B_{27}E_7 + B_{35}^T I_3 + B_{85}^T J_8\right) \\ & - B_{37}E_7 - L_{36}B_{86}^T\dot{J}_8 - B_{36}L_{66}B_{86}^T\dot{J}_8 \ . \end{aligned} \tag{5.69}$$

Betrachtet man nur die Koeffizienten von $V_4, \dot{V}_4, I_3$ und $\dot{I}_3$, lassen sich die Gleichungen (5.67) und (5.69) als Matrix wie folgt schreiben:

$$\begin{bmatrix} \dot{V}_4 \\ \dot{I}_3 \end{bmatrix} = \begin{bmatrix} M_C & 0 \\ 0 & M_L \end{bmatrix}^{-1} \begin{bmatrix} (-B_{24}^T M_R^{-1} B_{24}) & (B_{34}^T - B_{24}^T M_R^{-1} B_{25} R_{55} B_{35}^T) \\ (B_{35}M_G^{-1}B_{25}^T G_{22}B_{24} - B_{34}) & (-B_{35}M_G^{-1}B_{35}^T) \end{bmatrix} \begin{bmatrix} V_4 \\ I_3 \end{bmatrix} . \tag{5.70}$$

Dabei bedeuten

$$M_C = C_{44} + B_{14}^T C_{11} B_{14}$$

und

$$M_L = L_{33} + L_{36}B_{36}^T + B_{36}L_{63} + B_{36}L_{66}B_{36}^T \ .$$

Die Matrix auf der rechten Seite von Gleichung (5.70) hat jetzt die Form, wie sie im ersten Ausdruck auf der rechten Seite von Gleichung (5.65) benötigt wird. Sie stellt die allgemeine Form der A-Matrix in symbolischer Form dar.

Statt der symbolischen Formulierung kann SCEPTRE die Jacobi Matrix auch auf rein numerischen Weg berechnen. Dazu werden das Unterprogramm `SIMUL8` m mal aufgerufen, um die Ableitungen

$$\dot{Y}^{(j)} = F(Y_1, Y_2, \dots, Y_{j-1}, Y_j + \Delta Y_j, Y_{j+1}, \dots, Y_m, t)$$

zu berechnen. Die partiellen Ableitungen können dann durch folgende Näherung ermittelt werden:

$$\frac{\partial \dot{Y}_i}{\partial Y_j} \approx \frac{\dot{Y}_i^{(j)} - \dot{Y}_i}{\Delta Y_j} \,.$$

5.4 DC Analyse

SCEPTRE bietet 5 Lösungsverfahren für die DC-Analyse an: Berechnung der Anfangsbedingungen, Empfindlichkeit, Monte-Carlo, *Worst-Case* und Optimierung. Jede Analyseart kann für sich oder als Startwert für eine TR- oder AC-Analyse verwendet werden. Alternativ können die Anfangsbedingungen für eine TR- oder AC-Analyse vom Benutzer direkt vorgegeben werden. Alle DC-Analysen gehen von der Berechnung der Anfangsbedingungen aus und benutzen die Ergebnisse für ihre eigenen Berechnungen. Die Anfangsbedingungen — und somit auch die anderen Lösungsverfahren — können entweder mit der *Newton-Raphson* oder der *impliziten* Methode bestimmt werden.

5.4.1 Die Newton-Raphson Methode

Verfahren

Für viele praktische Schaltungen haben die Anfangsbedingungen einen entscheidenden Einfluss auf den Verlauf der TR-Analyse und müssen zu Beginn der TR-Analyse (Zeitpunkt t_0) berechnet werden. Eine Möglichkeit besteht darin, eine zusätzliche TR-Analyse durchzuführen, bei der alle Quellen konstant auf ihren Wert zum Zeitpunkt t_0 gehalten werden. Es gibt jedoch auch eine iterative Methode, bei der nicht die Zeit als unabhängige Variable benutzt wird. Diese Iteration erweist sich als vorteilhaft in Bezug auf die Rechenzeit bei Netzwerken, bei denen Konvergenz eintritt. Dieses Verfahren ist vollständig unabhängig von den Verfahren der TR-Analyse und wird im folgenden beschrieben.

Betrachtet man einen L-Baum[5], erhält man eine allgemeine B-Matrix nach Tabelle 5.4, bei der die entsprechenden Untermatrizen wegen der Eigenschaften des L-Baumes Null sind[6]. Unter der Annahme, dass die Vektoren V_6, I_1, I_4 sowie die Untermatrix B_{94} Null sind, lassen sich die folgenden Gleichungen aus dieser B-Matrix ableiten.

$$V_9 + B_{95}V_5 + B_{97}E_7 = 0 \tag{5.71}$$
$$V_2 + B_{25}V_5 + B_{27}E_7 = 0 \tag{5.72}$$
$$I_5 - B_{25}^T I_2 - B_{85}^T J_8 - B_{95}^T J_9 - B_{05}^T J_0 = 0 \tag{5.73}$$

[5] enthält die Netzwerkelemente in der bevorzugten Reihenfolge E, L, R und C

[6] Die Untermatrix B_{24} muss z. B. immer Null sein, da wegen der bevorzugten Reihenfolge keine Masche existiert mit einem Widerstand in einem Verbindungszweig und einer Kapazität in einem Baumzweig.

Tabelle 5.4: B-Matrix eines L-Baumes

	4	5	6	7
1	B_{14}	B_{15}	B_{16}	B_{17}
2	0	B_{25}	B_{26}	B_{27}
3	0	0	B_{36}	B_{37}
8	B_{84}	B_{85}	B_{86}	B_{87}
9	B_{94}	B_{95}	B_{96}	B_{97}
0	B_{04}	B_{05}	B_{06}	B_{07}

Um alle Variablen der Gleichung (5.73) durch die Vektoren V_9, V_2, V_5 sowie durch die vorhandenen Quellen ersetzen zu können, werden folgende Substitutionen gewählt:

$$\begin{aligned} I_5 &= G_{55}V_5 \\ I_2 &= G_{22}V_5 r \\ J_0 &= \alpha J_9 \\ J_9 &= G_{99}V_9 + Q_9 \ . \end{aligned}$$

Dabei sind

- α eine Matrix, deren Zeilenzahl der Anzahl der Stromquellen 2. Ordnung und deren Spaltenzahl der Anzahl der Stromquellen 1. Ordnung entsprechen;
- G_{55} und G_{22} Diagonalmatrizen, die nur Leitwerte enthalten;
- G_{99} eine Diagonalmatrix, die die Leitwerte von Dioden und Transistoren enthält.
- Die Q Terme werden im Anhang C.4 beschrieben.

Damit ergibt sich für Gleichung (5.73)

$$G_{55}V_5 - B_{25}^T G_{22}V_2 - B_{85}^T J_8 - \left[B_{95}^T + B_{05}^T\alpha\right]\left[G_{99}V_9 + Q_9\right] = 0 \ . \tag{5.74}$$

Die Gleichungen (5.71), (5.72) und (5.74) können auch als $F_1(V_9, V_2, V_5)$, $F_2(V_9, V_2, V_5)$ und $F_3(V_9, V_2, V_5)$ bezeichnet werden. Wird die Newton-Raphson Methode auf die Gleichungen (5.71), (5.72) und (5.74) angewendet, ergeben sich

$$\begin{aligned} F_1(V_9,V_2,V_5) + \frac{\partial F_1}{\partial V_9}(V_9,V_2,V_5)\Delta V_9 + \frac{\partial F_1}{\partial V_2}(V_9,V_2,V_5)\Delta V_2 + \frac{\partial F_1}{\partial V_5}(V_9,V_2,V_5)\Delta V_5 = 0 \\ F_2(V_9,V_2,V_5) + \frac{\partial F_2}{\partial V_9}(V_9,V_2,V_5)\Delta V_9 + \frac{\partial F_2}{\partial V_2}(V_9,V_2,V_5)\Delta V_2 + \frac{\partial F_2}{\partial V_5}(V_9,V_2,V_5)\Delta V_5 = 0 \\ F_3(V_9,V_2,V_5) + \frac{\partial F_3}{\partial V_9}(V_9,V_2,V_5)\Delta V_9 + \frac{\partial F_3}{\partial V_2}(V_9,V_2,V_5)\Delta V_2 + \frac{\partial F_3}{\partial V_5}(V_9,V_2,V_5)\Delta V_5 = 0 \end{aligned}$$

oder

$$\begin{bmatrix} F_1(V_9,V_2,V_5) \\ F_2(V_9,V_2,V_5) \\ F_3(V_9,V_2,V_5) \end{bmatrix} + Z \begin{bmatrix} \Delta V_9 \\ \Delta V_2 \\ \Delta V_5 \end{bmatrix} = 0 \tag{5.75}$$

mit der Jacobi Matrix

$$Z = \begin{bmatrix} \frac{\partial F_1}{\partial V_9}(V_9,V_2,V_5) & \frac{\partial F_1}{\partial V_2}(V_9,V_2,V_5) & \frac{\partial F_1}{\partial V_5}(V_9,V_2,V_5) \\ \frac{\partial F_2}{\partial V_9}(V_9,V_2,V_5) & \frac{\partial F_2}{\partial V_2}(V_9,V_2,V_5) & \frac{\partial F_2}{\partial V_5}(V_9,V_2,V_5) \\ \frac{\partial F_3}{\partial V_9}(V_9,V_2,V_5) & \frac{\partial F_3}{\partial V_2}(V_9,V_2,V_5) & \frac{\partial F_3}{\partial V_5}(V_9,V_2,V_5) \end{bmatrix} \tag{5.76}$$

. Mit n als Zähler der Iterationsschritte erhält man

$$\begin{aligned} \Delta V_9 &= V_9(n+1) - V_9(n) \\ \Delta V_2 &= V_2(n+1) - V_2(n) \\ \Delta V_5 &= V_5(n+1) - V_5(n) \end{aligned}$$

und damit aus Gleichung (5.75)

$$\begin{bmatrix} F_1(V_9,V_2,V_5) \\ F_2(V_9,V_2,V_5) \\ F_3(V_9,V_2,V_5) \end{bmatrix} + Z \begin{bmatrix} V_9(n+1) - V_9(n) \\ V_2(n+1) - V_2(n) \\ V_5(n+1) - V_5(n) \end{bmatrix} = 0 \tag{5.77}$$

und daraus schließlich

$$\begin{bmatrix} V_9(n+1) \\ V_2(n+1) \\ V_5(n+1) \end{bmatrix} = \begin{bmatrix} V_9(n) \\ V_2(n) \\ V_5(n) \end{bmatrix} - Z^{-1} \begin{bmatrix} F_1(V_9,V_2,V_5) \\ F_2(V_9,V_2,V_5) \\ F_3(V_9,V_2,V_5) \end{bmatrix} . \tag{5.78}$$

Gleichung (5.76) kann nach der Differenzierung der Gleichungen (5.71), (5.72) und (5.74) explizit geschrieben werden als

$$Z = \begin{bmatrix} I & 0 & B_{95} \\ 0 & I & B_{25} \\ -\left[B_{95}^T + B_{05}^T \alpha\right] G_{99} & -B_{25}^T G_{22} & G_{55} \end{bmatrix}$$

so dass man für Gleichung (5.78) erhält

$$\begin{bmatrix} V_9(n+1) \\ V_2(n+1) \\ V_5(n+1) \end{bmatrix} = \begin{bmatrix} V_9(n) \\ V_2(n) \\ V_5(n) \end{bmatrix} - \begin{bmatrix} Z \end{bmatrix}^{-1} \begin{bmatrix} F_1(V_9,V_2,V_5) \\ F_2(V_9,V_2,V_5) \\ F_3(V_9,V_2,V_5) \end{bmatrix} \tag{5.79}$$

oder in einer bequemeren Schreibweise (siehe Anhang C.7)

$$\begin{bmatrix} V_9(n+1) \\ V_2(n+1) \\ V_5(n+1) \end{bmatrix} = \begin{bmatrix} I & 0 & B_{95} \\ 0 & I & B_{25} \\ -\left[B_{95}^T + B_{05}^T \alpha\right] G_{99} & -B_{25}^T G_{22} & G_{55} \end{bmatrix}^{-1} \begin{bmatrix} -B_{97} E_7 \\ -B_{27} E_7 \\ \left[B_{95}^T + B_{05}^T \alpha\right] Q_9 + B_{85}^T J_8 \end{bmatrix} . \tag{5.80}$$

Gleichung (5.80) wird in der folgenden Reihenfolge berechnet: die Vektoren V_9, V_2 und V_5 besitzen zu Beginn der Iteration ($n = 0$) irgendwelche Werte, normalerweise Null. Alle Glieder auf der rechten Seite von Gleichung (5.80), die spannungsabhängig sind, werden aktualisiert. Anschließend wird die linke Seite ausgerechnet und der 1. Iterationsschritt ist fertig ($n = 1$). Danach wird die rechte Seite mit den Ergebnis der 1. Iteration bestimmt und danach wieder die linke Seite ($n = 2$). Diese Prozedur

wird bis zu 100 mal wiederholt. Nach jeder Iteration wird die Konvergenz für alle Spannungen in Gleichung (5.79) überprüft und die Iteration beendet, wenn gilt

$$|V_{n+1} - V_n| \leq |V_{n+1}| + 0.0001 \; . \tag{5.81}$$

Wird nach der 100. Iteration keine Konvergenz erreicht, druckt SCEPTRE eine entsprechende Nachricht. Die Erfahrung hat gezeigt, dass das Newton-Raphson-Verfahren im allgemeinen bei den meisten Netzwerken innerhalb von 30 Zyklen konvergiert.

Die Konvergenz des Newton-Raphson-Verfahrens verschlechtert sich drastisch, wenn an Dioden- oder Transistorübergängen hohe Durchlassspannungen anliegen und die Diodenstromquelle in der konventionellen Weise $J = I_s(e^{\Theta V} - 1)$ beschrieben wird. In diesem Fall ergibt sich aus der Steigung der Diodenkennlinie $G_J = \dfrac{dJ}{dV} = \Theta I_s e^{\Theta V}$ ein großer Wert für G_{99} und damit in der Z Matrix. Das führt zu einer sehr kleinen Integrationsschrittweite Δx der unabhängigen Variablen. Entsprechend werden sehr viele Integrationsschritte zur Konvergenz benötigt.

Sind die Vektoren V_9, V_2 und V_5 bekannt, können die Kapazitätsspannungen und Induktivitätsströme berechnet werden. Die Kapazitätsströme und Induktivitätsspannungen müssen Null sein.

Berechnung der Kapazitätsspannungen

Ausgehend von Tabelle 5.4 lassen sich die Spannungen an den Kapazitäten in den Verbindungszweigen durch Spannungen der Baumzweige ausdrücken:

$$V_1 = -B_{14}V_4 - B_{15}V_5 - B_{17}E_7 \; . \tag{5.82}$$

Aus dem Erhaltungssatz der elektrischen Ladung erhält man

$$-B_{14}^T C_{11} V_1 + C_{44} V_4 = -B_{14}^T C_{11} V_1(0) + C_{44} V_4(0) \tag{5.83}$$

wobei $V_1(0)$ und $V_4(0)$ vom Benutzer angegebene Anfangsspannungen sind. Aus (5.82) und (5.83) ergibt sich

$$V_4 = M_c^{-1} \left\{ -B_{14}^T C_{11} B_{15} V_5 - B_{14}^T C_{11} B_{17} E_7 - B_{14}^T C_{11} V_1(0) + C_{44} V_4(0) \right\} \tag{5.84}$$

mit $M_c = B_{14}^T C_{11} B_{14} + C_{44}$. Aus V_4 läßt sich V_1 mittels Gleichung (5.82) bestimmen.

Es sei angemerkt, dass Kapazitäten in Baumzweigen (Klasse 4) nur dann auftreten können, wenn eine Kapazitätsschnittmenge existiert. Andernfalls ist Gleichung (5.84) Null und man erhält die Kapazitätsspannungen direkt aus Gleichung (5.82).

Die Anwendung von Gleichung (5.83) erlaubt auch Schaltungen zu berechnen, bei denen die endgültige Spannung an einer Kapazität von einem Anfangswert abhängt. Die Schaltung nach Abbildung 5.6 enthält eine Kapazitätsschnittmenge. Wenn beide Kondensatoren ungeladen sind, stellen sich nach

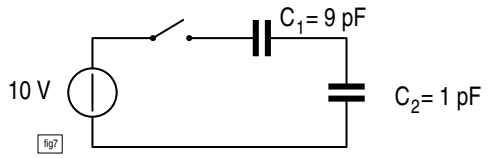


Abbildung 5.6: Schaltung mit Kapazitätsschnittmenge

Schließen des Schalters die Kondensatorspannungen $V_{C1} = 1\,\mathrm{V}$ und $V_{C2} = 9\,\mathrm{V}$ ein. Wenn jedoch der Kondensator C_1 eine Anfangsspannung von 5 V besitzt, müssen sich gemäß des Erhaltungssatzes der elektrischen Ladung (siehe Gleichung (5.83)) die Spannungen $V_{C1} = 5.5\,\mathrm{V}$ und $V_{C2} = 4.5\,\mathrm{V}$ einstellen.

Berechnung der Induktivitätsströme

Da V_2 bekannt ist, ergibt sich

$$I_2 = R_{22}^{-1} V_2 \tag{5.85}$$

und mit den Strömen in den Verbindungszweigen entsprechend der B-Matrix

$$I_6 = B_{26}^T I_2 + B_{36}^T I_3 + B_{86}^T J_8 + B_{96}^T J_9 + B_{06}^T J_0 \; . \tag{5.86}$$

Wegen der Erhaltung der magnetischen Energie gilt

$$B_{36} L_{66} I_6 + L_{33} I_3 = B_{36} L_{66} I_6(0) + L_{33} I_3(0) \tag{5.87}$$

mit $I_6(0)$ und $I_3(0)$ als Anfangsströme. Aus den Gleichungen (5.86) und (5.87) erhält man

$$I_3 = M_l^{-1} \left\{ L_{33} I_3(0) + B_{36} L_{66} I_6(0) - B_{36} L_{66} \left[B_{26}^T I_2 + B_{86}^T J_8 + B_{96}^T J_9 + B_{06}^T J_0 \right] \right\} \tag{5.88}$$

mit $M_l = B_{36} L_{66} B_{36}^T + L_{33}$. Ist I_3 bekannt, findet man I_6 mittels Gleichung (5.86).

Induktivitäten in Verbindungszweigen (Klasse 3) treten nur dann auf, wenn eine Induktivitätsschnittmenge existiert. Andernfalls ist Gleichung (5.88) Null und Gleichung (5.86) genügt zur Berechnung der Induktivitätsströme.

Die Einführung der Gleichung (5.88) gestattet es, Netzwerke zu berechnen, bei denen die endgültigen Ströme durch Induktivitäten von der Größe der jeweiligen Induktivität abhängen.

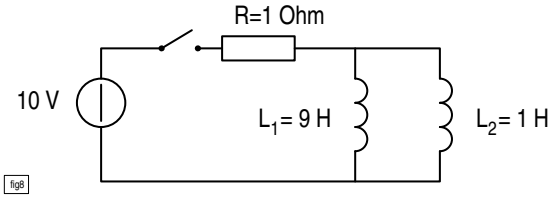


Abbildung 5.7: Schaltung mit einer Induktivitätsmasche

Die Schaltung 5.7 enthält eine Induktivitätsmasche. Sind beide Induktivitäten stromlos, müssen sich nach Schließen des Schalters die Ströme $I_{L1} = 1A$ und $I_{L2} = 9A$ einstellen.

Einschränkungen

Da die Berechnung der Anfangsbedingungen sich von der Berechnung der transienten Lösung unterscheidet, gelten für die Berechnung der Anfangsbedingungen Beschränkungen, die nicht für die Berechnung der transienten Lösung zutreffen. Dazu zählen:

- Es sind keine Maschen erlaubt, die nur aus Spannungsquellen und Induktivitäten bestehen. Die Folge wäre ein unendlich großer Strom durch die Induktivität.
- Es sind keine Knotenpunkte erlaubt, die nur Stromquellen und Kapazitäten miteinander verbinden (siehe Abbildung 5.8). Diese Situation würde Gleichung (5.71) verletzen und das Berechnungsverfahren erschweren, da auch noch V_4 bei dem Newton-Raphson-Verfahren berücksichtigt werden müßte. Durch Einfügen eines großen Widerstandes zwischen den Knoten `(A)` und `(0)` läßt sich dieses Problem beheben. Diese Einschränkung gilt nicht bei einer reinen TR-Analyse.

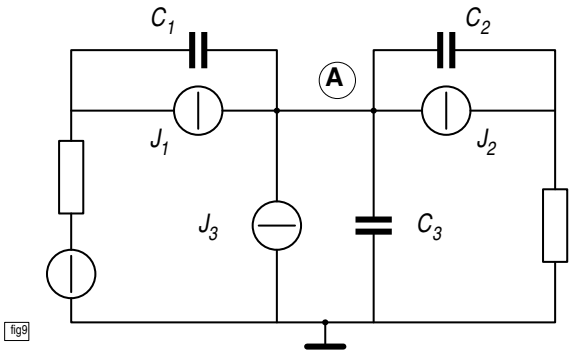


Abbildung 5.8: Knotenpunkt mit Kapazitäten und Stromquellen

- Als unabhängige Variable dürfen keine Ströme durch Induktivitäten oder Widerstände auftreten. Spannungen an Kondensatoren dürfen nur dann als unabhängige Variablen benutzt werden, wenn parallel zum Kondensator ein Widerstand ($V_C = V_R$) oder eine Stromquelle ($V_C = V_J$) liegt.
- Netzwerke, die aus Knotenpunkten bestehen, die nur Kapazitäten miteinander verbinden, sind nur dann erlaubt, wenn die Kapazitäten alle konstant sind.
- Netzwerke, die nur Maschen mit Induktivitäten enthalten, sind nur dann erlaubt, wenn alle Induktivitäten konstant sind.

5.4.2 Monte-Carlo

Die Monte-Carlo-Analyse führt mehrere Berechnungen der Anfangsbedingungen durch, wobei die Werte ausgesuchter Netzwerkelemente Toleranzen aufweisen müssen. Diese Abweichungen vom Nominalwert, die z. B. herstellerbedingt sein können, verursachen ebenso Abweichungen im Netzwerkverhalten. Die Monte-Carlo-Analyse liefert Statistiken über das Verhalten der vom Benutzer ausgesuchten Netzwerkgrößen und Elemente. Den mit Toleranzgrenzen versehenen Elementen werden dabei zufällige Werte innerhalb ihrer Toleranzgrenzen zugewiesen. Die Zufallswerte können gleichmäßig oder Gaußnormal verteilt sein. Da die Berechnung zur Zeit $t = 0$ erfolgt, haben die kapazitiven und induktiven Elemente keinen Einfluss auf das Ergebnis, d. h. es können nur Widerstände, unabhängige Strom- und Spannungsquellen sowie gesteuerte Stromquellen 1. und 2. Ordnung unter Benutzung definierter Parameter verwendet werden.

Der Wert eines Elements wird bestimmt aus dem Mittelwert μ, der Abweichung σ und einer Zufallszahl ζ. Der Zufallszahlengenerator liefert aus der gewählten Verteilung Proben mit dem Mittelwert 0 und der Standardabweichung 1. Für den Wert eines Elementes gilt also

$$element = \mu + \zeta \cdot \sigma \ .$$

Falls die zur Verfügung stehenden Zufallszahlenverteilungen (Gleich-, Gaußsche Normalverteilung) nicht geeignet sind, können definierte Parameter zur Beschreibung komplizierterer Verteilungen benutzt werden. Die Netzwerkelemente werden dann mit den definierten Parametern in Verbindung von arithmetischen Anweisungen und Tabellen beschrieben.

Bei jedem Durchgang werden den Elementen mit Toleranzangaben Werte entsprechend der spezifizierten Zufallszahlenverteilung zugewiesen und die Anfangsbedingungen ausgerechnet. Für jede

Ausgangsgröße werden Nominalwert, beobachtetes Maximum und Minimum, Mittelwert und Standardabweichung ausgegeben. Der Mittelwert M_x erhält man aus

$$M_x = \frac{\sum_i x_i}{N},$$

für die Standardabweichung S_x gilt

$$S_x = \sqrt{\left[\frac{\sum_i (x_i - M_x)^2}{N-1}\right]}$$

wobei N die Anzahl der gewünschten Monte-Carlo Durchgänge ist.

5.4.3 Empfindlichkeit

Die Empfindlichkeitsberechnung (*sensitivity*) liefert die partielle Ableitung einer Netzwerkfunktion H in Abhängigkeit anderer unabhängiger Netzwerkvariablen X. Ausgegeben wird die „absolute“ Empfindlichkeit (*partial derivative, un-normalized sensitivity*)

$$\int_X^{UH} \frac{\partial H}{\partial X}$$

und die normalisierte Empfindlichkeit (*normalized sensitivity*)

$$\int_X^{H} \frac{X}{H}\frac{\partial H}{\partial X}.$$

Zur Berechnung der Empfindlichkeit werden zunächst die Anfangsbedingungen des ursprünglichen Netzwerkes berechnet. Dann wird ein adjungiertes Netzwerk erzeugt und für jede gewünschte Variable die Anfangsbedingungen in diesem Netzwerk ausgerechnet. Aus dem Produkt dieser Ströme und Spannungen mit den entsprechenden Werten aus dem Originalnetzwerk erhält man die gewünschte Empfindlichkeit. Tabelle 5.5 zeigt die Berechnungsvorschrift für die entsprechenden unabhängigen Variablen (der Index „a“ kennzeichnet Größen des adjungierten Netzwerkes).

Tabelle 5.5: Empfindlichkeitsberechnung

unabhängige Variable	Empfindlichkeit
Widerstand R	(Strom durch Widerstand im Originalnetzwerk) * (Strom durch den gleichen Widerstand im adjungierten Netzwerk), also $I_R \cdot I_{Ra}$
unabhängige Spannungsquelle E_7	negativer Strom durch die gleiche Spannungsquelle im adjungierten Netzwerk, also $-I_{E7a}$
unabhängige Stromquelle J_8	negative Spannung an der gleichen Stromquelle im adjungierten Netzwerk, also $-V_{J8a}$
Stromquelle 2. Ordnung J_0 mit definiertem Parameter P, also $J_0 = P \cdot J_9$	(Stromquelle 1. Ordnung im Originalnetzwerk) * (negative Spannung an der Stromquelle 2. Ordnung im adjungierten Netzwerk), also $J_9 \cdot (-V_{J0a})$

5.4.4 *Worst-Case*

Zur Bestimmung des „ungünstigsten Falles“ wird der Gradient der Zielfunktion F linear extrapoliert bis zu seinem Minimum oder Maximum an den Grenzen einer oder mehrerer unabhängiger Variablen. Der Gradientenvektor ∇F wird aus der Nominalanalyse des Netzwerkes gewonnen. Dabei wird angenommen, dass die Funktion F linear abhängig ist von den N unabhängigen Variablen. Der Gradient gibt die Richtung an, in der die Zielfunktion den größten Zuwachs besitzt. Durch die angenommene Linearität kann durch Extrapolation entlang von F das jeweilige Maximum oder Minimum bestimmt werden.

So erhält man das Maximum der Zielfunktion durch Extrapolation der unabhängigen Variablen X innerhalb der Toleranzgrenzen entlang von F aus

$$X_H = Y + \lambda \nabla F, \quad \lambda > 0 \; . \tag{5.89}$$

X_H enthält die unabhängigen Variablen, die zum Maximum von F führen und Y stellt die Nominalwerte der unabhängigen Variablen dar. λ wird dabei so groß gewählt, dass X_H innerhalb der Toleranzgrenzen liegt.

Für das Minimum gilt mit dem Faktor μ analog

$$X_L = Y - \mu \nabla F, \quad \mu > 0 \; . \tag{5.90}$$

Die Toleranzintervalle der unabhängigen Variablen sind definiert als

$$P_i \leq X_i \leq Q_i \; , \qquad i = 1 \ldots N.$$

Das bedeutet, dass bei positivem Gradient $\partial F / \partial X_i$ die Funktion F wächst, wenn X_i von Y_i in Richtung Q_i zunimmt. Entsprechend wird F bei negativem Gradient kleiner, wenn X_i von Y_i in Richtung P_i abnimmt.

Eine Gerade im N dimensionalen Raum wird durch die allgemeine Gleichung

$$\frac{X_1 - Y_1}{\mu_1} = \frac{X_2 - Y_2}{\mu_2} = \ldots = \frac{X_N - Y_N}{\mu_N}$$

beschrieben mit $X_i, Y_i, i = 1 \ldots N$ als Geradenpunkte und μ_i als Einheitsvektoren, d. h.

$$\frac{X_i - Y_i}{\mu_i} = \text{constant} \; . \tag{5.91}$$

Für die j. Komponente an der unteren Toleranzgrenze gilt

$$\frac{X_i - Y_i}{\mu_i} = \frac{P_j - Y_j}{\mu_j}, \quad i = 1 \ldots N$$

oder

$$X_i = \frac{\mu_i}{\mu_j}(P_j - Y_j) + Y_i \; .$$

Die Länge des Geradenstückes ergibt sich aus

$$D^2_{P_j} = \sum_i (X_i - Y_i)^2 = \frac{(P_j - Y_j)^2}{\mu_j^2} \sum_i \mu_i^2 = \frac{(P_j - Y_j)^2}{\mu_j^2}$$

und somit beträgt der Abstand längs des Gradientenvektors zur unteren Toleranzgrenze

$$D_{P_j} = \left| \frac{P_j - Y_j}{\mu_j} \right| .$$

Entsprechend erhält man für den Abstand zur oberen Toleranzgrenze

$$D_{Q_j} = \left| \frac{Q_j - Y_j}{\mu_j} \right| .$$

Zur Bestimmung des höchsten Wertes werden alle D_{P_i}, für die $\partial F/\partial X_i > 0$, und alle D_{Q_i}, für die $\partial F/\partial X_i < 0$, nach dem kleinsten Wert D_{high} durchsucht[7]. D_{high} stellt die größte Entfernung dar, in der sich die unabhängigen Variablen X von Y entlang des Gradientenvektors bewegen können, bevor sie eine Toleranzgrenze erreichen. Dabei ist X_{high} die entsprechende unabhängige Variable, die zu D_{high} gehört.

Nach dem gleichen Verfahren werden alle D_{P_i}, für die $\partial F/\partial X_i < 0$, und alle D_{Q_i}, für die $\partial F/\partial X_i > 0$, nach dem kleinsten Wert D_{low} durchsucht, der den größtmöglichsten Abstand angibt, in den sich X von Y in negativer Richtung entlang des Gradientvektors bewegen kann. Zu diesem Wert gehört die unabhängige Variable X_{low}.

X_{low} und X_{high} können, müssen aber nicht die gleiche Variable darstellen. Abbildung 5.9 zeigt ein zweidimensionales Beispiel, bei dem der höchste Wert durch die obere Toleranzgrenze der Variablen X_2 und der niedrigste Wert ebenfalls durch die obere Toleranzgrenze einer anderen Variablen X_1 bestimmt wird.

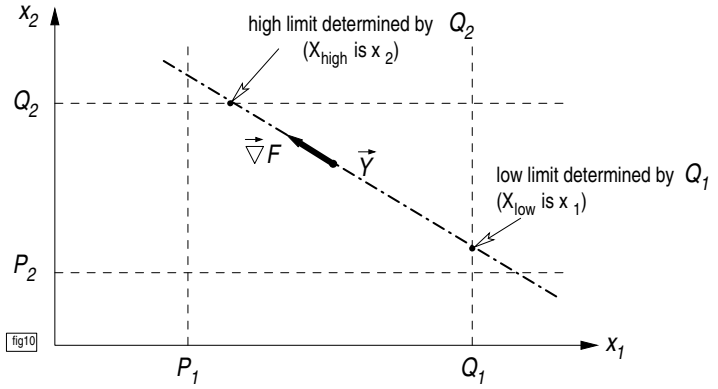


Abbildung 5.9: Zweidimensionales Beispiel zur *Worst-Case*-Analyse

Ist D_{high} bekannt, können mittels Gleichung (5.91) alle anderen unabhängigen Variablen X_H berechnet werden, die zu dem Maximum der Zielfunktion gehören:

$$\frac{X_i - Y_i}{\mu_i} = D_{high}$$

oder

$$X_H = Y + \mu D_{high}.$$

Diese Gleichung stimmt exakt mit Gleichung (5.89) überein, da μ proportional zu ∇F und $D_{high} > 0$ ist.

[7] Die Funktion F ist unabhängig von jedem X_i, für das gilt $\partial F/\partial X_i = 0$.

In gleicher Weise findet man die unabhängigen Variablen X_L für das Minimum der Zielfunktion

$$X_L = Y - \mu D_{low}$$

entsprechend Gleichung (5.90).

Sind X_L und X_H bekannt, werden die Zielfunktionen und Gradienten an jeden der beiden Punkten ausgerechnet. Der Gradientenvektor an diesen Punkten liefert Aufschluss darüber, ob die tatsächlichen Extremwerte innerhalb oder außerhalb der Toleranzgrenzen liegen.

5.4.5 Optimierung

Die Optimierung innerhalb der DC-Analyse von SCEPTRE findet das lokale Minimum einer Zielfunktion mit Rücksicht auf unabhängige Variablen mit Toleranzgrenzen. Ein lokales Minimum liegt dann vor, wenn entweder der Gradient Null ist, oder negativ und die zugehörige unabhängige Variable an einer Toleranzgrenze liegt.

Die Optimierung — oder besser gesagt die Minimierung — beruht auf dem *Davidon*-Verfahren [11]. Dabei wird für einen Anfangswert der unabhängigen Variablen Y der Gradient innerhalb der Toleranzgrenzen berechnet und Y entlang des Gradientenvektors verschoben. An diesem Punkt wird erneut der Gradientenvektor berechnet. Dieser Vorgang wiederholt sich solange, bis eine vorgegebene Fehlerschranke unterschritten wird.

Die Berechnung des Gradienten mit Hilfe der adjungierten Netzwerkmethode vermeidet die Verwendung direkter Suchmethoden mit den dazugehörenden numerischen Problemen. Außerdem erhält man den Gradienten mit Hilfe der adjungierten Methode aus zwei Anfangsbedingungsrechnungen unabhängig von der Anzahl der unabhängigen Variablen, was ein weiterer Zeitvorteil gegenüber der direkten Suchmethode oder anderen Gradientenverfahren bedeutet. Die Rechenzeit verringert sich zusätzlich dadurch, dass die Berechnung der Anfangsbedingungen für das adjungierte Netzwerk linear ist und somit schnell konvergiert.

Die Davidon Methode setzt sich aus zwei Verfahren zusammen: die Methode des Steilsten Abstiegs und der Newton-Raphson-Algorithmus, angewendet auf die Berechnung der Anfangsbedingungen (siehe Abschnitt 5.4.1, Seite 101).

Der Newton-Raphson-Algorithmus geht von den ersten beiden Gliedern der Taylorreihe aus:

$$F(X) = F(Y) + h \cdot \nabla F(Y) + \frac{1}{2} h \cdot G(Y) h$$

mit h als Verschiebungsvektor $(X - Y)$ und G als *Hesss*che Matrix der zweiten partiellen Ableitungen der Zielfunktion

$$G_{ij} = \frac{\partial^2 F}{\partial X_i \partial X_j} \,. \tag{5.92}$$

Leitet man diesen Asudruck nach $F(X)$ ab, erhält man den Gradientenvektor

$$\nabla F(X) = \nabla F(Y) + G(Y) h.$$

Da für den kleinsten Wert von X gilt $\nabla F(X) = 0$, ergibt sich für den Verschiebungsvektor

$$h = -\frac{1}{G(Y)} \nabla F(Y).$$

In unmittelbarer Nähe des Minimums wird eine quadratische Abhängigkeit zwischen F und X angenommen, sodass G an diesem Punkt ohne Schwierigkeiten bestimmt werden kann.

Die Methode des steilsten Abstiegs orientiert sich vollständig an dem Gradientenvektor und besticht durch seine Einfachheit. Bei Annäherung an das Minimum verringert sich jedoch die Konvergenz drastisch, sodass man dieses Verfahren nur zu Beginn des Optimierungsprozesses benutzt.

Das Davidon Verfahren wechselt allmählich von der Methode des steilsten Abstiegs zum Newton-Raphson-Algorithmus. Die Kombination beider Verfahren stellt somit einen sehr effizienten Suchprozess dar [12].

Das Davidon Verfahren [13] beginnt mit einer Anfangsnäherung H_0 für G^{-1}, im allgemeinen der Einheitsmatrix. Diese ist positiv definit, da die Eigenwerte alle Eins sind. Diese Eigenschaft bleibt während der Iteration erhalten, bei der ohne Matrixinversion eine zunehmende Annäherung von H für G^{-1} eintritt.

Mit (i) als Iterationsschritt, $X^{(i)}$ als unabhängige Variablen, $h^{(i)}$ als Schrittweite, die mittels eines linearen Suchalgorithmus bestimmt wird, und $H^{(i)}$ als neue Näherung läßt sich das Davidon Verfahren beschreiben

$$X^{(i+1)} = X^{(i)} + h^{(i)} H^{(i)} \nabla F^{(i)} \ .$$

Die Neuberechnung von H geschieht folgendermaßen:

$$\begin{aligned} Z &= -h^{(i)} \nabla F^{(i)} \\ \mu &= \nabla F^{(i+1)} - \nabla F^{(i)} \\ H^{(i+1)} &= H^{(i)} + A^{(i)} + B^{(i)} \ . \end{aligned}$$

Z und μ stellen Vektoren, A und B Matrizen dar, deren Koeffizienten gegeben sind durch

$$\begin{aligned} a_{jk}^{(i)} &= \frac{Z_j Z_k}{Z \cdot \mu} \\ b_{jk}^{(i)} &= \frac{(H^{(i)}\mu) j (H^{(i)}\mu) k}{\mu \cdot H\mu} \ . \end{aligned}$$

Die Matrix A trägt zur Konvergenz von H bei, während B sicherstellt, dass H positiv definit bleibt.

Das Originalverfahren [14] wurde für SCEPTRE entsprechend [12] abgeändert, um den Nebenbedingungen der unabhängigen Variablen in SCEPTRE gerecht zu werden. Zu diesem Zweck müssen die unabhängigen Variablen transformiert werden.

Für die unabhängige Variable X_i gilt: $P_i \leq X_i \leq Q_i$. Dann gilt für die Transformation

$$X_i = \frac{Q_i + P_i}{2} - \frac{Q_i - P_i}{2} \cos Y_i$$

und deren Ableitung

$$\frac{dX_i}{dY_i} = \frac{Q_i - P_i}{2} \sin Y_i \ .$$

An den Toleranzgrenzen gelten

$$\begin{aligned} X_i = P_i &\rightsquigarrow Y_i = 0 \\ X_i = Q_i &\rightsquigarrow Y_i = \pi \\ \frac{dX_i}{dY_i} &= 0 \ . \end{aligned}$$

Für die Komponenten des Gradientenvektors bezüglich der neuen Variablen ergibt sich

$$\frac{\partial F}{\partial Y_i} = \frac{dX_i}{dY_i}\frac{\partial F}{\partial X_i} \; .$$

Durch diese Transformation ist der Gradient an den Toleranzgrenzen Null und innerhalb der Toleranzgrenzen $\neq$ Null.

Weiterhin muss beim Originalverfahren das Minimum der Zielfunktion positiv sein. In SCEPTRE wird ein variabler Referenzwert eingeführt, der kleiner als der geschätzte Minimalwert der Funktion ist und der automatisch der Iteration angepaßt wird. Der Benutzer kann diesen Wert selber vorgeben und damit die Konvergenz beschleunigen, bei falscher Vorgabe aber auch verschlechtern.

Die Iteration wird beendet, wenn die erwartete Änderung der Zielfunktion ΔF beim nächsten Zuwachs des unabhängigen Variablenvektors $X^{(i)}$ nach $X^{(i+1)}$ kleiner als eine vorgegebene Abbruchsgrenze ist (siehe Abbildung 5.10). Als Minimum wird der Wert von $X^{(i+1)}$ ausgegeben.

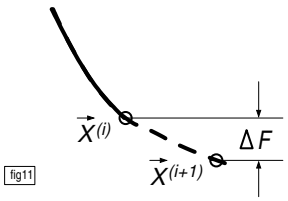


Abbildung 5.10: Änderung der Zielfunktion bei Änderung der unabhängigen Variablen

5.5 AC Analyse

Die folgende Beschreibung der Wechselstromanalyse (AC) unterscheidet vier Fälle:

1. Schaltungen mit unabhängigen Quellen (Kapitel 5.5.1)
   $\dot{X} = AX + GV$, mit $V^T = [E_7\ J_8]$
2. Schaltungen mit linear abhängigen Quellen (Kapitel 5.5.2)
   $\dot{X} = AX + GV + Hu$, mit $u^T = [E_Y\ J_X]$
3. Schaltungen, die die zeitlichen Ableitungen unabhängiger Quellen benötigen (Kapitel 5.5.3)
   $\dot{X} = AX + GV + Q\dot{V}$
4. Schaltungen der Fälle 2 und 3 (Kapitel 5.5.4)
   $\dot{X} = AX + GV + Q\dot{V} + Hu$

5.5.1 Fall 1: nur unabhängige Quellen

Zustandsvariablen

Ein lineares, zeit-invariantes System für eine Frequenzanalyse lßt sich durch folgendes Gleichungssystem beschreiben:

$$\dot{X} = AX + GV \tag{5.93}$$
$$Y = CX + DV \tag{5.94}$$

mit
A als $(N \times N)$ Matrix mit reellen Koeffizienten, die das Netzwerk beschreibt;
G als $(N \times N_S)$ Matrix mit reellen Koeffizienten, die die externen Quellen des Netzwerks beschreibt;
N als Anzahl der Zustandsvariablen;
N_S als Anzahl der externen Quellen;
N_E als Gesamtanzahl der Netzwerkelemente;.
N_R als Anzahl der Ausgabegrößen; es gilt $N_R \leq 2 \cdot N_E$;
C als $(N_R \times N)$ Matrix mit reellen Koeffizienten, die die Zustandsvariablen als Ausgabegrößen zuordnet;
D als $(N_R \times N_S)$ Matrix mit reelen Koeffizienten, die die Eingabegrößen den Ausgabegrößen zuordnet;.
X als $(N \times 1)$ Vektor der Zustandsvariablen;
V als $(N_S \times 1)$ Vektor der externen Quellen;
Y als $(N_R \times 1)$ Vektor der gewünschten Ausgabegrößen.

Jede frequenzabhängige Anregungsfunktion (Quelle) kann man in der Form beschreiben

$$V_k(t,\omega) = K_k(\omega)\ \mathrm{e}^{\alpha t}\ , \qquad 1 \leq k \leq N_S \tag{5.95}$$

mit
ω als Frequenz in (rad),
α als $\alpha = \mathrm{j}\,\omega$,
$K_k(\omega)$ als komplexe Funktion der Frequenz,

oder in Vektorschreibweise

$$V = K\ \mathrm{e}^{\alpha t}\ . \tag{5.96}$$

Die allg. Lösung von Gleichung (5.93) lautet

$$X = X_0\ \mathrm{e}^{\alpha t}\ . \tag{5.97}$$

Mit den Gleichungen (5.96) und (5.97) ergibt sich für Gleichung (5.93)

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\,t}\,(X_0\ \mathrm{e}^{\alpha t}) &= A\,(X_0\ \mathrm{e}^{\alpha t}) + G\,(K\ \mathrm{e}^{\alpha t}) \\
\alpha X_0\ \mathrm{e}^{\alpha t} &= A X_0\ \mathrm{e}^{\alpha t} + GK\ \mathrm{e}^{\alpha t} \qquad \text{und} \\
\alpha X_0 &= A X_0 + GK \qquad \text{oder} \\
(\alpha I - A) X_0 &= GK\ .
\end{aligned}$$

Aufgelöst nach X_0 erhält man

$$X_0 = (\alpha I - A)^{-1} GK\ . \tag{5.98}$$

Die Matrix A wird mit dem Unterprogramm `EIGENP` [15, 16] auf eine Diagonalmatrix reduziert:

$$A = S\ \Lambda\ S^{-1} \tag{5.99}$$

mit
Λ als $(N \times N)$ Diagonalmatrix mit komplexen Eigenwerten,
S als komplexe $(N \times N)$ Matrix, dessen Spalten die Eigenvektoren darstellen.
Diese Matrix wird auch als *Modal Matrix* bezeichnet.

Setzt man (5.99) in (5.98) ein erhält man

$$\begin{aligned} X_0 &= (S\alpha S^{-1} - S\Lambda S^{-1})^{-1} GK \\ X_0 &= S(\alpha I - \Lambda)^{-1} S^{-1} GK \qquad \text{oder} \\ X_0 &= S(\mathrm{j}\,\omega I - \Lambda)^{-1} S^{-1} GK(\omega)\ . \end{aligned} \tag{5.100}$$

Gleichung (5.100) stellt die AC Lösung der Zustandsvariablengleichung (5.93) dar.

Gleichung (5.94) beschreibt die Variablen, die keine Zustandsvariablen sind. Die AC Lösung für diese Ströme und Spannungen entsprechen denen der transienten Rechnung (Kapitel 5.3.1) in komplexer Darstellung:

$$\begin{aligned} I_1 &= \mathrm{j}\,\omega C_{11} V_1 \\ I_4 &= \mathrm{j}\,\omega C_{44} V_4 \\ V_3 &= \mathrm{j}\,\omega (L_{36} I_6 + L_{33} I_3) \\ V_6 &= \mathrm{j}\,\omega (L_{36}^T I_3 + L_{66} I_6) \end{aligned}$$

A und *G* Matrix Berechnung

In SCEPTRE sind die Ströme I_3 durch Induktivitäten und die Spannung V_4 an Kapazitäten Zustandsvariablen. Weiterhin sind Spannungsquellen E_7 und Stromquellen J_8 erlaubt. Ausgedrückt durch X und V kann man schreiben

$$X = \left[\begin{array}{c} I_3 \\ V_4 \end{array} \right] \Bigg\} N\,, \qquad V = \left[\begin{array}{c} E_7 \\ J_8 \end{array} \right] \Bigg\} N_S\ .$$

und unter Berücksichtigung von (5.93) in Matrizenschreibweise

$$\left[\begin{array}{c} \dot{I}_3 \\ \dot{V}_4 \end{array} \right] = \left[\begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right] \left[\begin{array}{c} I_3 \\ V_4 \end{array} \right] + \left[\begin{array}{cc} G_{11} & G_{12} \\ G_{21} & G_{22} \end{array} \right] \left[\begin{array}{c} E_7 \\ J_8 \end{array} \right]\ ,$$

oder in Gleichungsform

$$\begin{aligned} \dot{I}_3 &= A_{11} I_3 + A_{12} V_4 + G_{11} E_7 + G_{12} J_8 \\ \dot{V}_4 &= A_{21} I_3 + A_{22} V_4 + G_{21} E_7 + G_{22} J_8\ . \end{aligned}$$

In Kapitel 5.3.1 [15, 16] wurden die transienten Gleichungen benutzt, um die notwendigen Untermatrizen abzuleiten. Zur Beschreibung der passiven und zeitinvarianten Elemente R, L und C ergeben sich die Untermatrizen zu

$$A_{11} = (-M_{\mathrm{LI}}) * B_{35} * R_{55} * [B_{35}^T - B_{25}^T * (M_{\mathrm{RI}}) * B_{25} * R_{55} * B_{35}^T] \tag{5.101}$$
$$A_{12} = (-M_{\mathrm{LI}}) * [B_{34} - B_{35} * R_{55} * B_{25}^T * (M_{\mathrm{RI}}) * B_{24}] \tag{5.102}$$
$$A_{21} = (C_{44} I) * [B_{14}^T * (M_{\mathrm{SI}}) * B_{14} * S_{44} - I] * [B_{24}^T * (M_{\mathrm{RI}}) * B_{25} * R_{55} * B_{35}^T - B_{34}^T] \tag{5.103}$$
$$A_{22} = (C_{44} I) * [B_{14}^T * (M_{\mathrm{SI}}) * B_{14} * S_{44} - I] * [B_{24}^T * (M_{\mathrm{RI}}) * B_{24}] \tag{5.104}$$

mit

$$\begin{aligned} M_{\mathrm{LI}} &= [L_{33} + L_{36} * B_{36}^T + B_{36} * L_{63} + B_{36} * L_{66} * B_{36}^T]^{-1} \\ M_{\mathrm{RI}} &= [R_{22} + B_{25} * R_{55} * B_{25}^T]^{-1} \\ M_{\mathrm{SI}} &= [S_{11} + B_{14} * S_{44} * B_{14}^T]^{-1}\ . \end{aligned}$$

Zur Beschreibung der unabhängigen Strom- und/oder Spannungsquellen ergeben sich die folgende Untermatrizen:

$$G_{11} = (-M_{\mathrm{LI}}) * [B_{37} - B_{35} * R_{55} * B_{25}^T * (M_{\mathrm{RI}}) * B_{27}] \tag{5.105}$$
$$G_{12} = (-M_{\mathrm{LI}}) * [B_{35} * R_{55} * (B_{85}^T - B_{25}^T * (M_{\mathrm{RI}}) * B_{25} * R_{55} * B_{85}^T] \tag{5.106}$$
$$G_{21} = (C_{44}I) * [(B_{14}^T * (M_{\mathrm{SI}}) * B_{14} * S_{44} - I) * B_{24}^T * (M_{\mathrm{RI}}) * B_{27}] \tag{5.107}$$
$$G_{22} = (C_{44}I) * [(B_{14}^T * (M_{\mathrm{SI}}) * B_{14} * S_{44} - I) * (B_{24}^T * (M_{\mathrm{RI}}) * B_{25} * R_{55} * B_{85}^T - B_{84}^T)] \ . \tag{5.108}$$

5.5.2 Fall 2: Schaltungen mit linearabhängigen Quellen

Reduzierung in die kanonische Form

In SCEPTRE sind spannungs- und stromgesteuerte Quellen, die linear von Widerstandspannungen und -strömen abhängen, als E_Y und J_X definiert. Gegeben sei ein Vektor u in der Form $u^T = [E_Y J_X]$. Dann lässt sich die Schaltung durch eine Gleichung

$$\dot{X} = AX + GV + Hu \tag{5.109}$$

beschreiben mit H als $(N \times N_D)$ Matrix mit reellen Koeffizienten, die die abhängigen Quellen den Zustandsvariablen zuordnet. N_D ist die Anzahl der linear abhängigen Quellen.

Um Gleichung (5.109) in die kanonische Form

$$\dot{X} = (A_{\mathrm{EXT}})X + (G_{\mathrm{EXT}})V$$

zu bringen mit (A_{EXT}) als $(N \times N)$ Matrix mit reellen Koeffizienten, die das Netzwerk beschreiben, und(G_{EXT}) als $(N \times N_S)$ Matrix mit reellen Koeffizienten zur Beschreibung der externen Quellen im Netzwerk, kann man Gleichung (5.109) folgendermaßen umschreiben:

$$\begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} E_Y \\ J_X \end{bmatrix} \ . \tag{5.110}$$

Für die Quellen E_Y und J_X gilt

$$E_Y = k_1 V_2 + k_2 V_5 \tag{5.111}$$
$$J_X = k_3 I_2 + k_4 I_5 \tag{5.112}$$

mit k_1, k_2, k_3, k_4 als Matrizen, die die Abhängigkeit der Quellen zu den entsprechenden Widerstandsspannungen und -strömen beschreiben. Setzt man $V_2 = R_{22} I_2$ und $I_5 = G_{55} V_5$ in die Gleichungen (5.111) und (5.112) ein, erhält man:

$$\begin{bmatrix} E_Y \\ J_X \end{bmatrix} = \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} \begin{bmatrix} I_2 \\ V_5 \end{bmatrix} \ . \tag{5.113}$$

Gleichung (5.113) in Gleichung (5.110) ergibt dann

$$\begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} \begin{bmatrix} I_2 \\ V_5 \end{bmatrix} \ . \tag{5.114}$$

Der Vektor $[Y]^T = [I_2 V_5]$ lässt sich mittels Zustandsvariablen und unabhängigen Quellen ausdrücken wenn man berücksichtigt, dass Y' eine Untermenge von $Y = CX + DV$ darstellt, als

$$\begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = Y' = C'X + D'V. \tag{5.115}$$

Entsprechend Kapitel 5.3.4 gilt

$$\begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = M_{\mathrm{RG}}^{-1} \begin{bmatrix} -B_{24}V_4 & -B_{27}E_7 \\ B_{35}^T I_3 & +B_{85}^T J_8 \end{bmatrix}$$

mit

$$M_{\mathrm{RG}} = \begin{bmatrix} (R_{22} + B_{2Y}k_1R + 22) & (B_{25} + B_{2Y}k_2) \\ (-B_{25}^T - B_{X5}^T k_3) & (G_{55} - B_{X5}^T k_4 G_{55}) \end{bmatrix}$$

und weiterhin

$$C' = M_{\mathrm{RG}}^{-1} \begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \tag{5.116}$$

und

$$D' = M_{\mathrm{RG}}^{-1} \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix}. \tag{5.117}$$

Mit den Gleichungen (5.117), (5.116) und (5.115) kann man Gleichung (5.114) durch Zustandsvariablen und unabhängigen Quellen ausdrücken:

$$\begin{aligned} \begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} = &\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} \\ &+ \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1R_{22} & k_2 \\ k_3 & k_4G_{55} \end{bmatrix} M_{\mathrm{RG}}^{-1} \left(\begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} \right) \end{aligned}$$

oder

$$\begin{aligned} \begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} = &\left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1R_{22} & k_2 \\ k_3 & k_4G_{55} \end{bmatrix} M_{\mathrm{RG}}^{-1} \begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \right) \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} \\ &+ \left(\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1R_{22} & k_2 \\ k_3 & k_4G_{55} \end{bmatrix} M_{\mathrm{RG}}^{-1} \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix} \right) \begin{bmatrix} E_7 \\ J_8 \end{bmatrix}. \end{aligned} \tag{5.118}$$

Gleichung (5.118) stellt eine Umformung von Gleichung (5.109) in die kanonische Form dar mit

$$A_{\mathrm{EXT}} = \left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1R_{22} & k_2 \\ k_3 & k_4G_{55} \end{bmatrix} M_{\mathrm{RG}}^{-1} \begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \right)$$

und

$$G_{\mathrm{EXT}} = \left(\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1R_{22} & k_2 \\ k_3 & k_4G_{55} \end{bmatrix} M_{\mathrm{RG}}^{-1} \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix} \right).$$

Berechnung der H-Matrix

Gleichung (5.118) enthält die Matrix H, die in ihrem Aufbau der Matrix G ähnelt. Betrachtet man die Gleichung

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} E_Y \\ J_X \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & H_{11} & H_{12} \\ G_{21} & G_{22} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \\ E_Y \\ J_X \end{bmatrix}$$

erkennt man, dass sich die beiden Untermatrizen nur in den Termen, die sich auf die Quellen beziehen, unterscheiden, also

$$G = \left[\begin{array}{r|l} (g_1)B_{37} + (g_2)B_{27} & (g_4)B_{85}^T \\ +(g_3)B_{27} & (g_5)B_{85}^T + (g_6)B_{84}^T \end{array}\right]$$

und

$$H = \left[\begin{array}{r|l} (g_1)B_{3Y} + (g_2)B_{2Y} & (g_4)B_{X5}^T \\ +(g_3)B_{2Y} & (g_5)B_{X5}^T + (g_6)B_{X4}^T \end{array}\right]$$

mit

$g_1 = -M_{\mathrm{LI}}$
$g_2 = M_{\mathrm{LI}}(B_{35}R_{55})(B_{25}^T M_{\mathrm{RI}})$
$g_3 = C_{44}I(B_{14}^T M_{\mathrm{SI}} B_{14} S_{44} - I)B_{24}^T M_{\mathrm{RI}}$
$g_4 = -M_{\mathrm{LI}}(B_{35}R_{55}[I - (B_{25}^T M_{\mathrm{RI}})(B_{25}R_{55})]$
$g_5 = C_{44}I(B_{14}^T M_{\mathrm{SI}} B_{14} S_{44} - I)(B_{24}^T M_{\mathrm{RI}} B_{25} R_{55})$
$g_6 = -C_{44}I(B_{14}^T M_{\mathrm{SI}} B_{14} S_{44} - I).$

5.5.3 Fall 3: Schaltungen, die die zeitlichen Ableitungen unabhängiger Quellen benötigen

Herleitung der AC Lösung

Wenn Schaltungen für eine TR-Analyse die zeitliche Ableitung von Quellen benötigen, z. B. bei Maschen aus unabhängigen Spannungsquellen und Kapazitäten oder Knoten mit unabhängigen Stromquellen und Induktivitäten, so werden diese Ableitungen auch bei der AC-Analyse benötigt. Anders als bei der TR-Analyse muss der Benutzer die Ableitung nicht selber vorgeben sondern sie werden von SCEPTRE berechnet, und zwar wie folgt:

es gilt

$$\dot{X} = AX + GV + Q\dot{V} \qquad \text{mit } V_k(t,\omega) = K_k(\omega)\ \mathrm{e}^{\mathrm{j}\,\omega t} \tag{5.119}$$

und der Lösung

$$X = X_0\ \mathrm{e}^{\mathrm{j}\,\omega t}\ .$$

Daraus ergibt sich

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\left(X_0\ \mathrm{e}^{\mathrm{j}\,\omega t}\right) = A(X_0\ \mathrm{e}^{\mathrm{j}\,\omega t}) + G(K\ \mathrm{e}^{\mathrm{j}\,\omega t}) + Q\frac{\mathrm{d}}{\mathrm{d}\,t}(K\ \mathrm{e}^{\mathrm{j}\,\omega t})$$
$$\mathrm{j}\,\omega X_0\ \mathrm{e}^{\mathrm{j}\,\omega t} = AX_0\ \mathrm{e}^{\mathrm{j}\,\omega t} + GK\ \mathrm{e}^{\mathrm{j}\,\omega t} + \mathrm{j}\,\omega QK\ \mathrm{e}^{\mathrm{j}\,\omega t}$$

und

$$(\mathrm{j}\,\omega I - A)X_0 = GK + \mathrm{j}\,\omega QK$$
$$X_0 = (\mathrm{j}\,\omega I - A)^{-1}(GK + \mathrm{j}\,\omega QK)$$

und nach einer Ähnlichkeitstransformation schließlich

$$X_0 = S(\mathrm{j}\,\omega I - A)^{-1}S^{-1}(GK + \mathrm{j}\,\omega QK)$$
$$X_0 = S(\mathrm{j}\,\omega I - A)^{-1}[(S^{-1}GK) + \mathrm{j}\,\omega(S^{-1}QK)]\ . \tag{5.120}$$

Gleichung (5.120) stellt die AC-Lösung von Gleichung (5.119) dar. Der zweite Term in den Klammern berücksichtigt die Ableitung aller Quellen.

Berechnung der Matrix Q

Für die Matrix Q gilt

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$$

mit

$Q_{11} = 0$
$Q_{12} = (-M_{\mathrm{LI}})[B_{36}L_{66}B_{86}^T + L_{36}B_{86}^T]$
$Q_{21} = (-C_{44}I)B_{14}^T M_{\mathrm{SI}} B_{17}$
$Q_{22} = 0$.

5.5.4 Fall 4: Schaltungen mit linear abhängigen Quellen, die zeitliche Ableitungen unabhängiger Quellen benötigen

Hierbei handelt es sich um eine Kombination der Fälle 5.5.2 und 5.5.3.

6 Progammablauf

SCEPTRE wurde als FORTRAN IV Programm für den Rechner IBM 7090/94 und dem IBM System 360 (OS/360) geschrieben. Im Originalhandbuch Volume II [3] werden die einzelnen Programmschritte erläutert. Diese Erläuterungen werden in diesem Handbuch weggelassen. Eine Beschreibung des Programmablaufs unter Linux/UNIX findet der Leser im Kapitel 7.

Die folgenden Abbildungen zeigen den Programmablauf für die Rechner IBM 7090/94 und OS/360. Sie wurden aus dem Originalhandbuch unverändert übernommen.

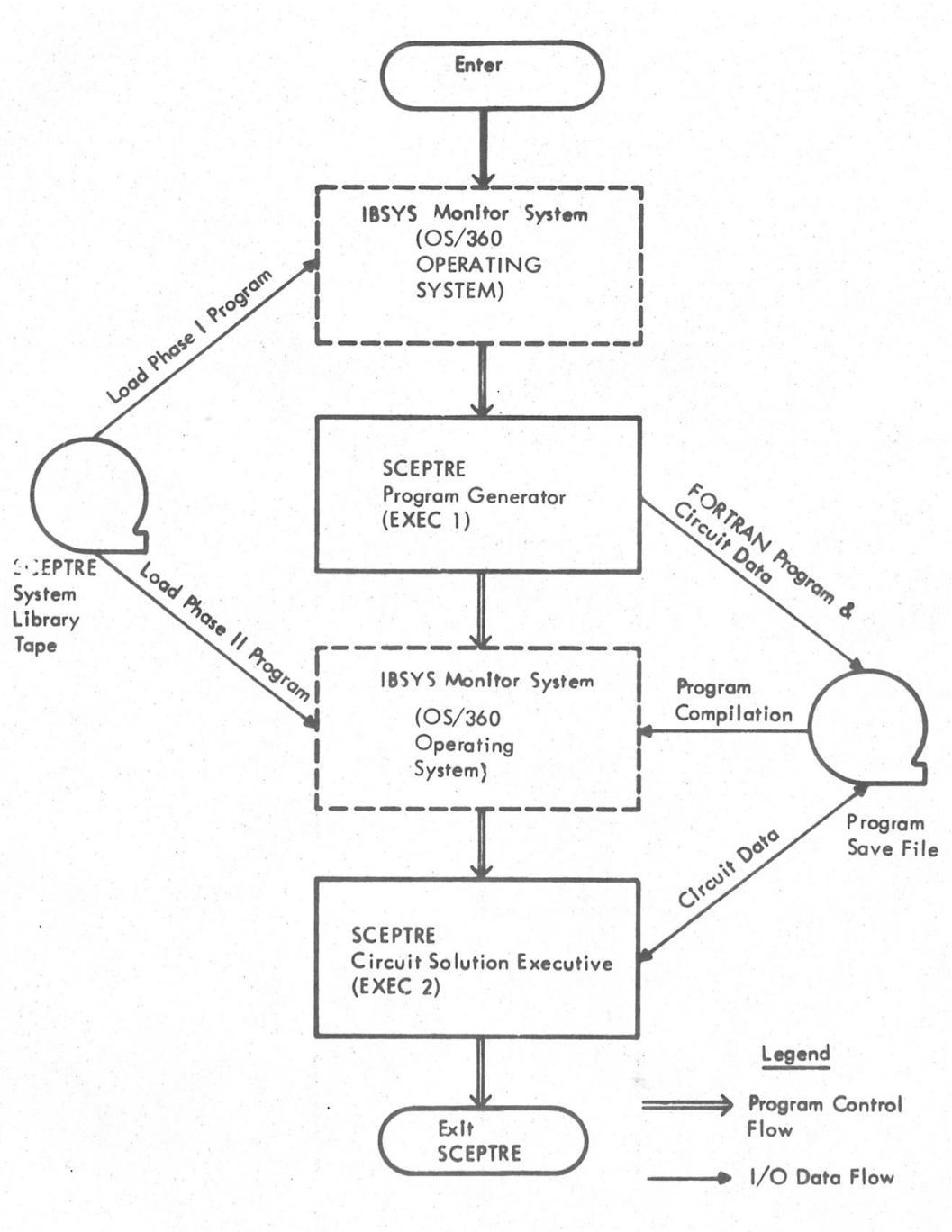


Abbildung 6.1: System Flow Diagram

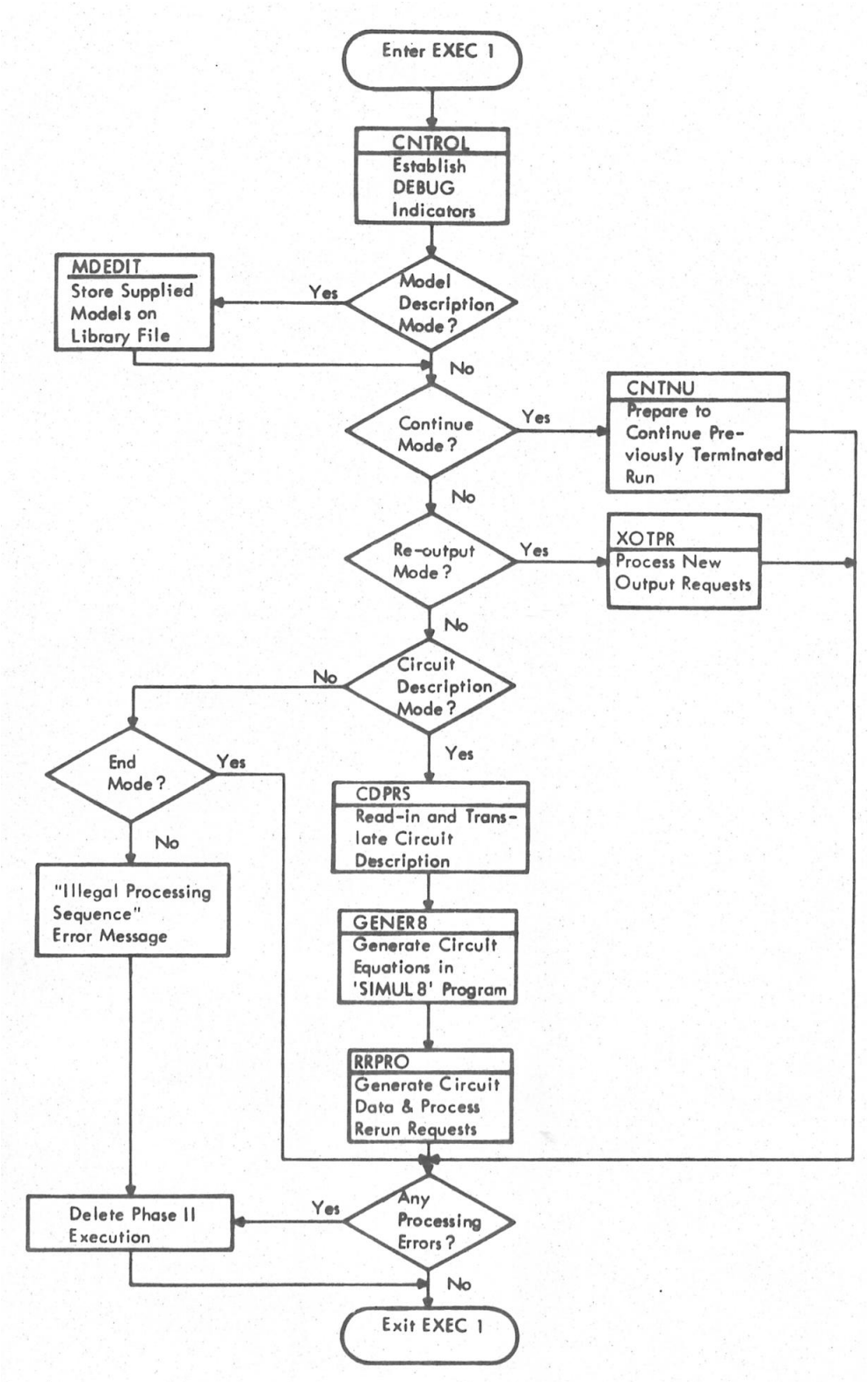


Abbildung 6.2: Program Generator (EXEC1) Flow Diagram

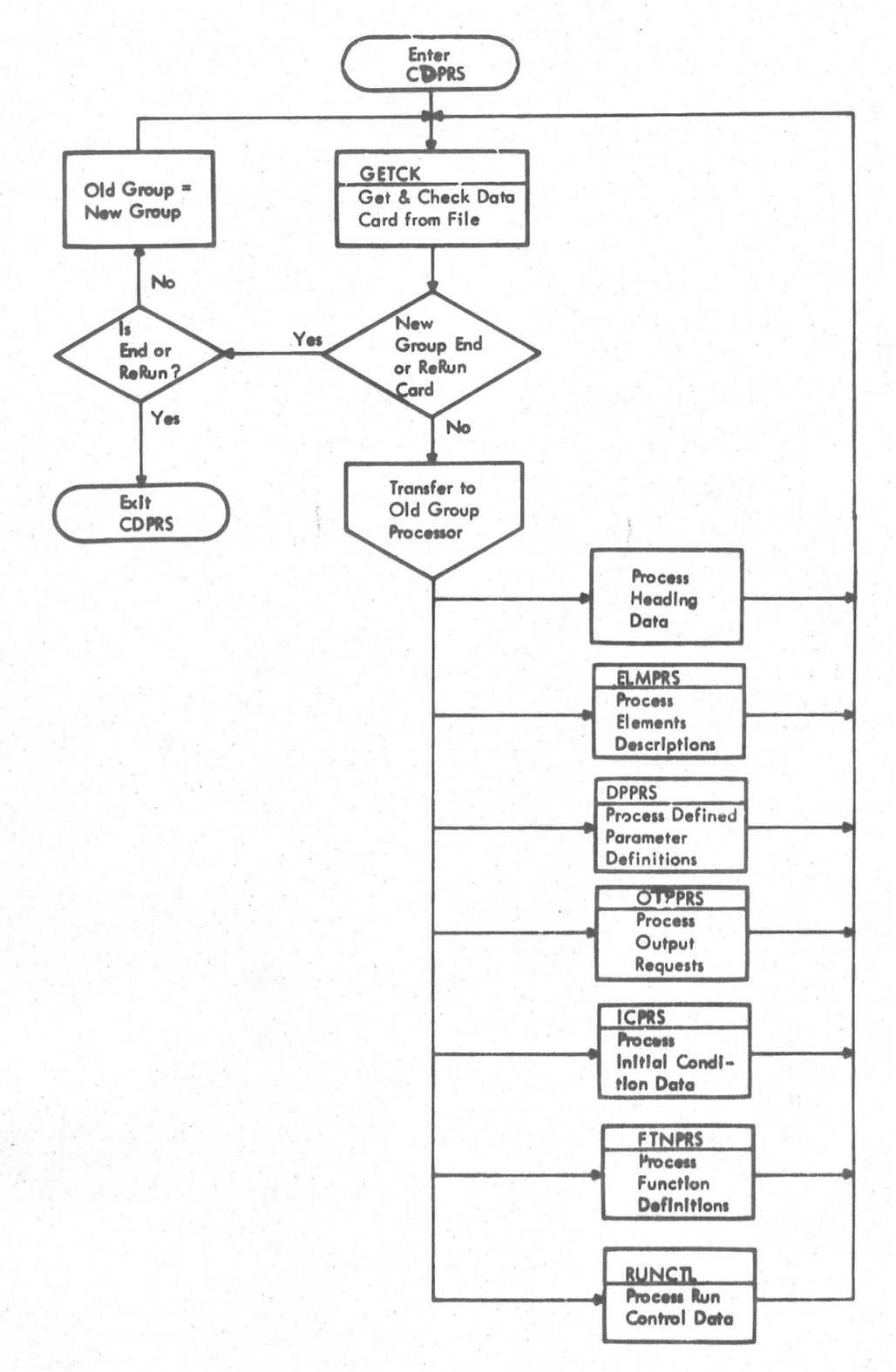


Abbildung 6.3: Circuit Description Processor (CDPRS) Flow Diagram

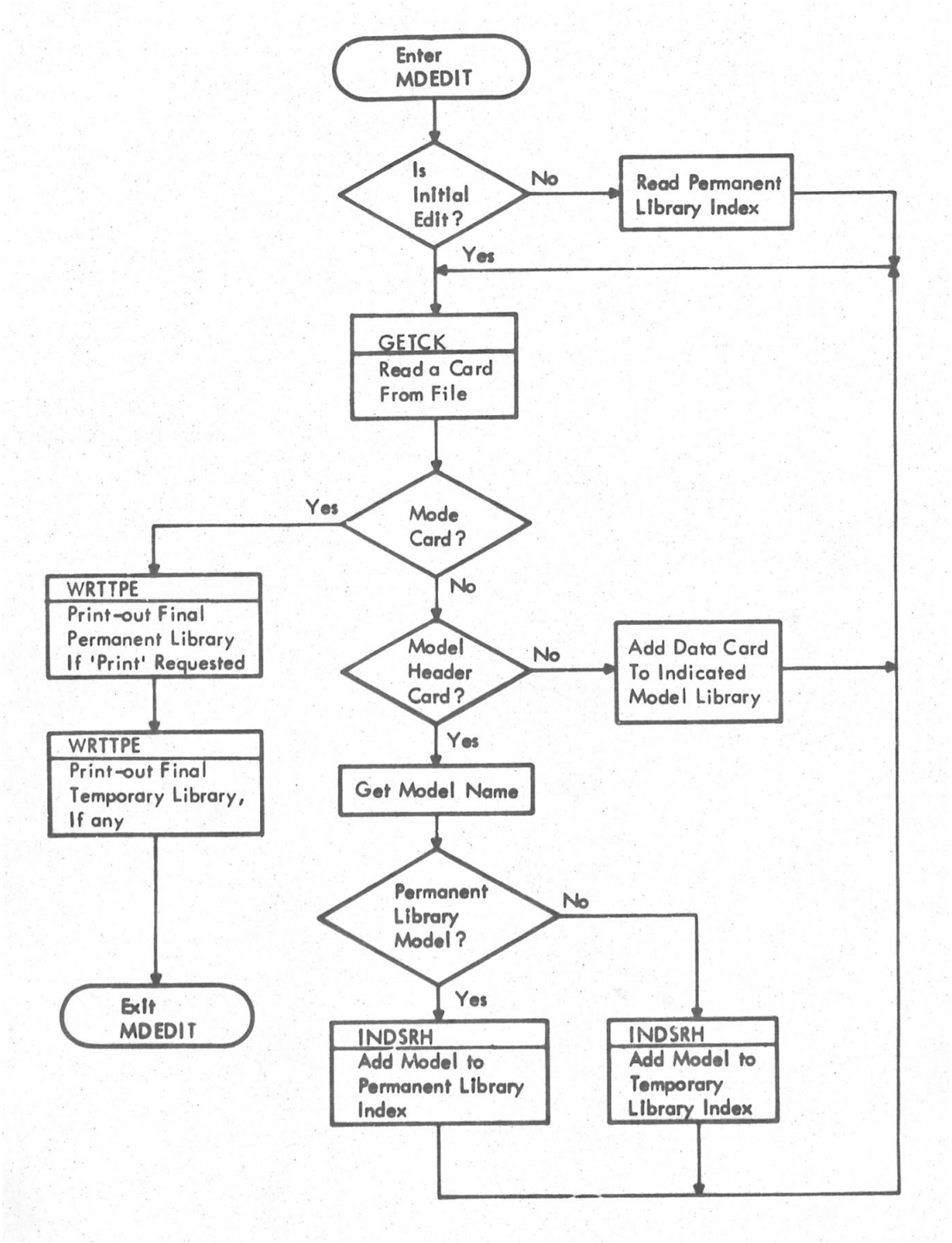


Abbildung 6.4: Model Editor (MDEDIT) Flow Diagram

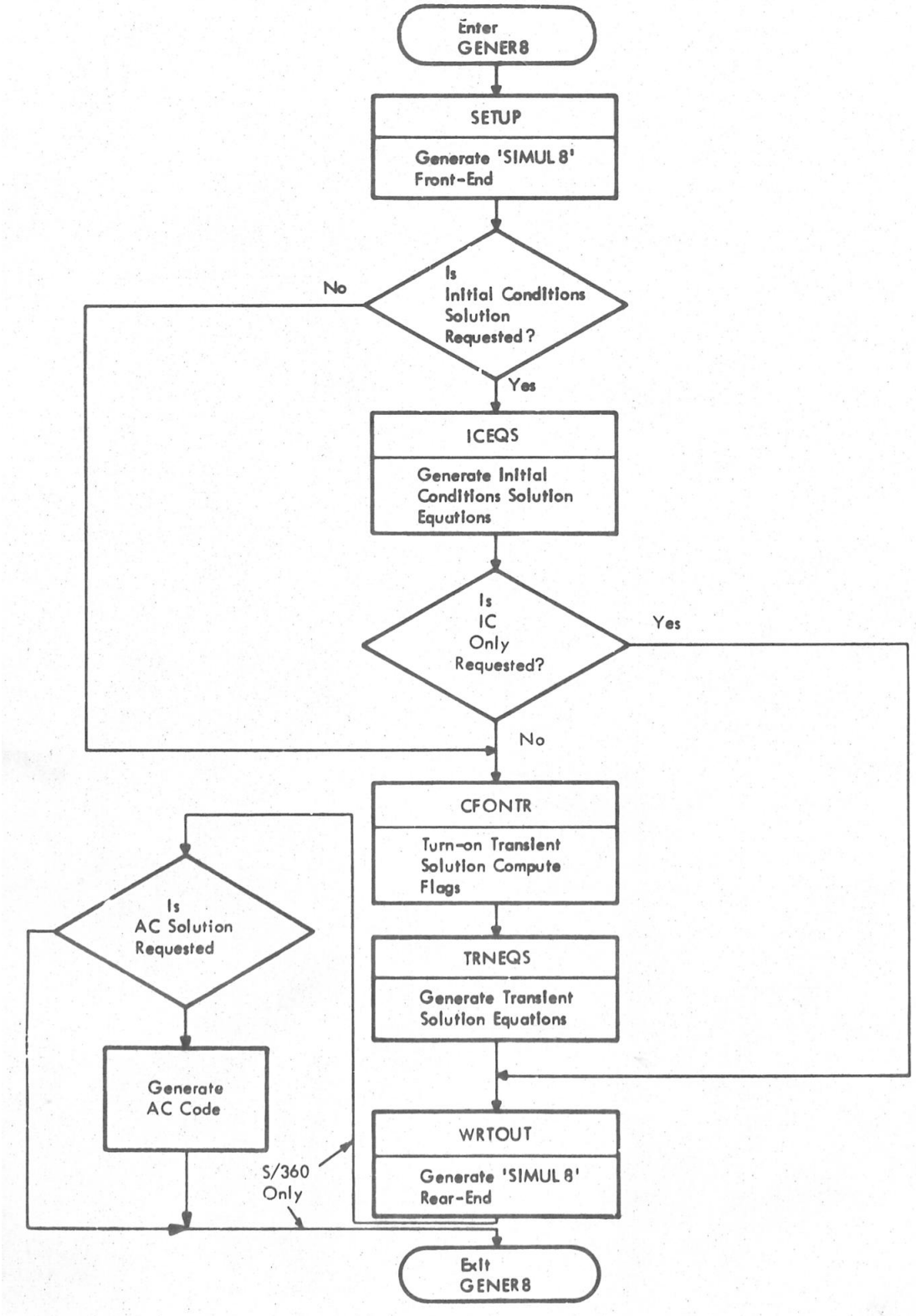


Abbildung 6.5: Circuit Equation Generator (GENER8) Flow Diagram

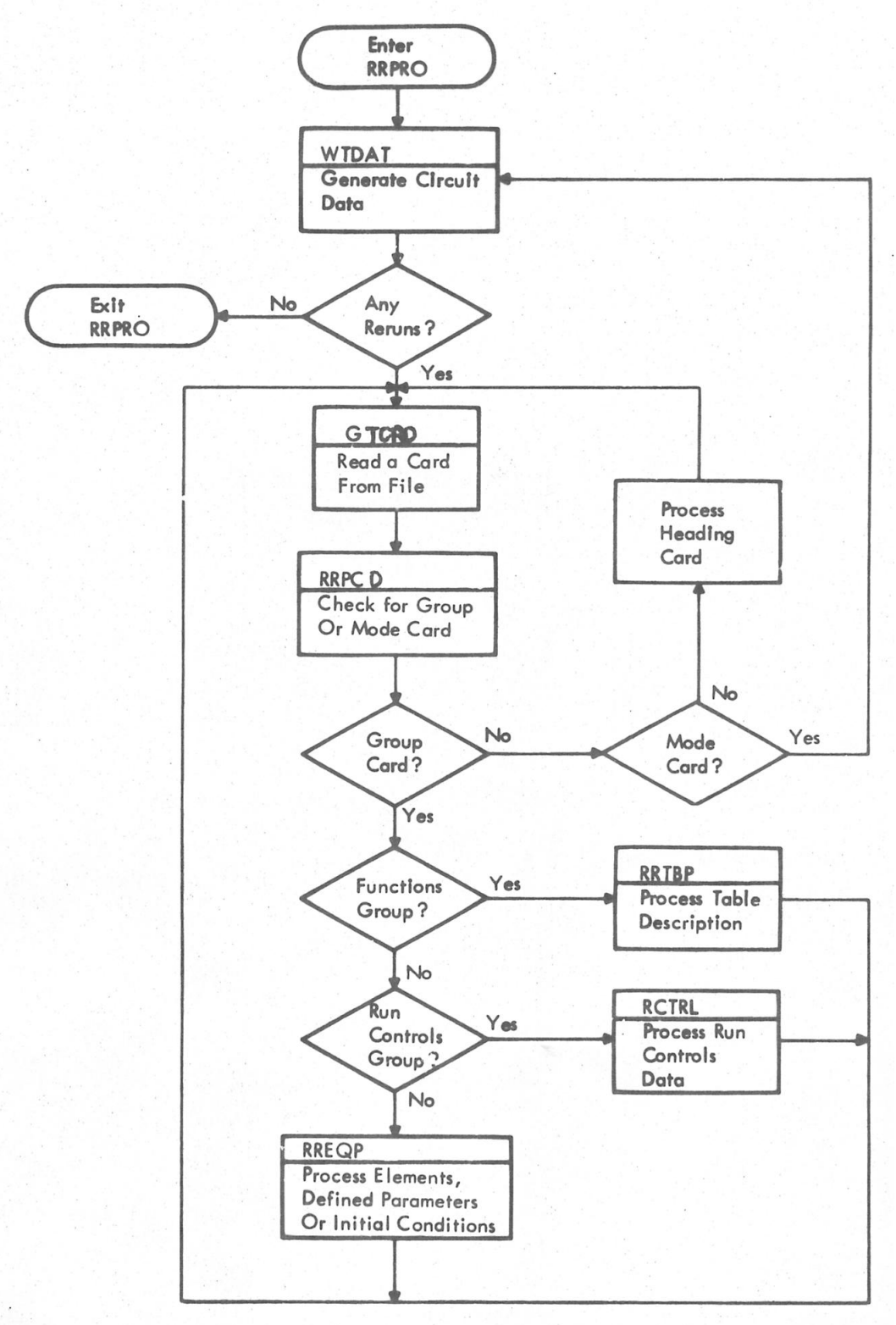


Abbildung 6.6: Rerun Processor and Data Generator (RRPRO) Flow Diagram

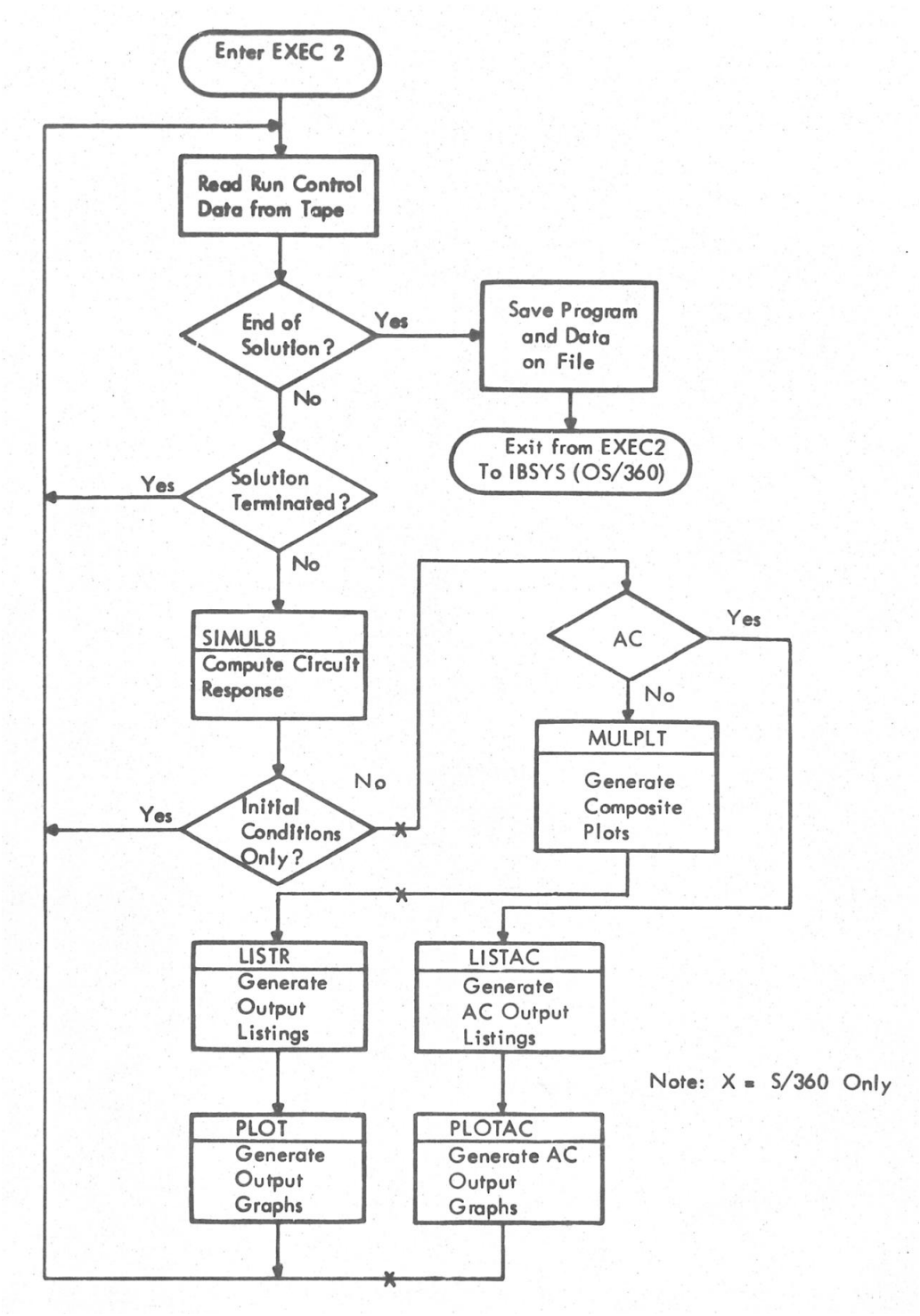


Abbildung 6.7: Circuit Solution Executive (EXEC2) Flow Diagram

Teil III

Benutzungsanleitung

7 Das Programmsystem SCEPTRE (Linux/UNIX-Version)

7.1 Programmorganisation

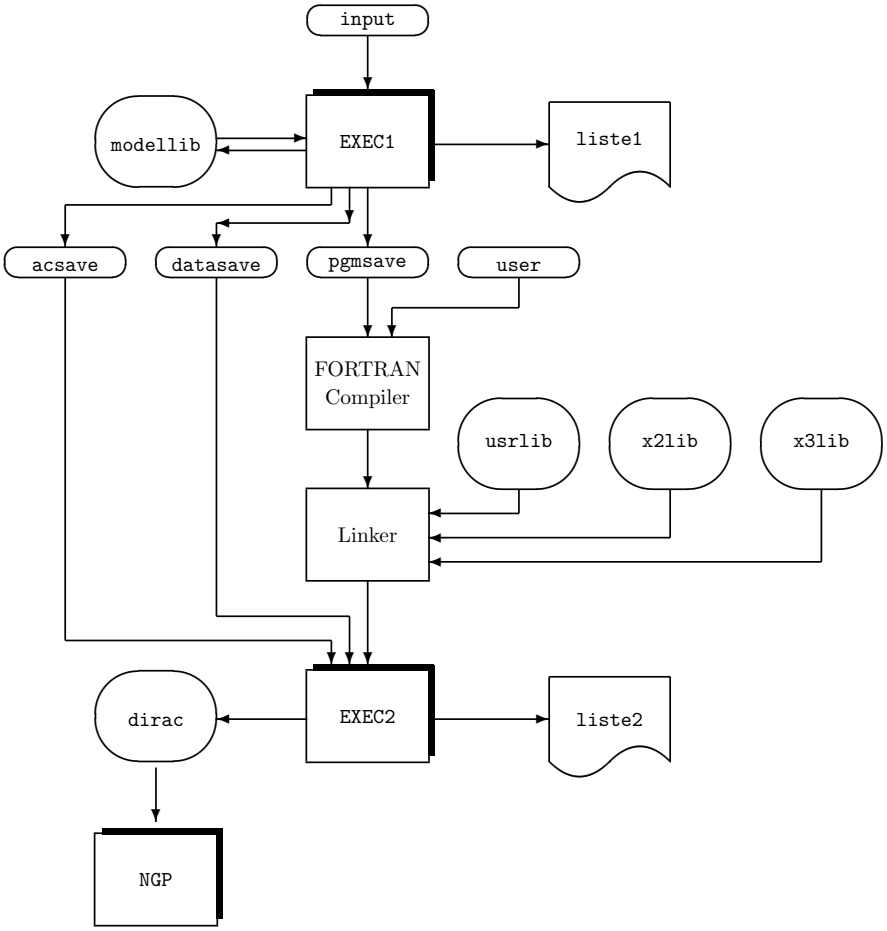


Abbildung 7.1: Programm- und Datenfluss

SCEPTRE läuft in mehreren Phasen ab (Abbildung 7.1):

1. Mit dem Programm `EXEC1` werden die Eingabedaten verarbeitet. Dabei können permanente Modelle in die Modellbibliothek abgespeichert oder aus der gleichen Bibliothek in die zu untersuchende Schaltung eingefügt werden. Sind die Eingabedaten fehlerfrei, werden die Dateien `acsave` und `datasave` für die Ausführungsphase und einige FORTRAN-Unterprogramme (`SIMUL8, SIMTR, LOUTR, ...`) für die Datei `pgmsave` generiert.

2. Die Unterprogramme in der Datei `pgmsave` werden, evtl. mit eigenen FORTRAN Unterprogrammen (`user`), übersetzt.

3. Der Linker bindet die Objektmodule des FORTRAN Übersetzers mit den vorübersetzen Programmen aus den Bibliotheken `usrlib, x2lib` und `x3lib` zu einem ausführbaren Programm `EXEC2` zusammen. Die Bibliotheken `x2lib` und `x3lib` gehören zu SCEPTRE, `usrlib` dient zur Aufnahme von vorübersetzten Benutzerunterprogrammen.

4. Das Programm `EXEC2` wird ausgeführt. Dazu werden die Dateien `datasave` und `acsave` aus der 1. Phase benötigt. Neben der Standarddruckausgabe `liste2` werden die Ergebnisse mit dem Unterprogramm `POST` (siehe Kapitel 7.4) zwecks Nachverarbeitung in eine direkte Zugriffsdatei `dirac` geschrieben.

Diese etwas umständlich erscheinende Programmorganisation hat einen entscheidenden Vorteil gegenüber direkt ausführbaren Programmen: das generierte Programm `EXEC2` ist auf das zu untersuchende Problem zugeschnitten. Alle Vektoren und Arbeitsbereiche sind nur so groß dimensioniert, wie sie für diese Simulation benötigt werden. Ebenso sind nur die Unterprogramme vorhanden, die durch die Analyseart bestimmt werden. Weiterhin ist es sehr einfach, eigene Programme beliebig in eine SCEPTRE Simulation einzubinden, was die Universalität von SCEPTRE erhöht.

7.2 Aufruf unter Unix

SCEPTRE wird mit dem Kommando

`sceptre` *dateiname* ohne Dateiendung

gestartet. Die SCEPTRE-Eingabedaten müssen in der Datei *dateiname*`.d` stehen.

Nach Verarbeitung der Eingabedaten wird das Unix-Kommando `less` benutzt, mit dem man sich die Druckausgabe der ersten Phase (*dateiname*`.ls1`) anschauen kann. `less` wird mit der Taste `Q` beendet. Sind die Eingabedaten fehlerfrei, wird automatisch zur zweiten Phase übergegangen.

Standardmäßig werden bei der AC- und TR-Analyse die *laufenden* Ergebnisse als „Digitalanzeige“ mittels des ANSI-Treibers ausgegeben (siehe Kapitel 7.3.2). Falls der Ausgabebildschirm nicht dazu geeignet ist oder die Ausgabe unterdrückt werden soll, muss in den SCEPTRE Eingabedaten unter der Überschrift `RUN CONTROLS` die Anweisung `X PLOT DIMENSION = 0` eingefügt werden.

Am Ende jeder Simulation und vor jedem neuen Wiederholungslauf muss die Simulation mit dem Befehl `go` fortgeführt werden (Bildschirmnachrichten des Systems beachten).

Anschließend wird wieder mit `less` die Druckausgabe der zweiten Phase (*dateiname*`.ls2`) angezeigt (Taste `Q` zum Beenden). Zum Schluß werden alle Dateien *dateiname*`.*` aufgelistet.

Siehe auch Anhang D.

Der Postprozessor NGP (siehe Kapitel 7.5) wird mit dem Kommando

`ngp` *dateiname*

unter X11 aufgerufen.

7.3 Grafikschnittstellen

Standardmäßig liefert SCEPTRE Druckerplots, und zwar nur am Ende eines jeden Simulationslaufes. Das Programm bietet jedoch zwei Schnittstellen, um externe Grafiksoftware benutzen und zusätzlich *während* der Simulation die Ergebnisse grafisch oder tabellarisch („Digitalanzeige“) darstellen zu können. Dazu müssen die beiden Platzhalterunterprogramme `POST` und `INTPLT` durch eigene Routinen ersetzt werden, zweckmäßigerweise in der Benutzerbibliothek `usrlib`, da diese Routinen *vor* den Unterprogrammbibliotheken `x2lib` und `x3lib` in der Phase 2 gelinkt werden müssen.

7.3.1 Das Unterprogramm `INTPLT`

Das Unterprogramm wird zu jedem Integrations- bzw. Frequenzschritt aufgerufen und beeinflusst entscheidend die Rechenzeit der Simulation. Aus diesem Grund werden die auszugebenden Daten nicht als Parameter im Aufruf übergeben, sondern es wird auf die internen COMMON-Blöcke zugegriffen. Aufruf:

```
1       subroutine INTPLT (nstep)
```

Die Variable `nstep` enthält die Nummer des aktuellen Integrations- bzw. Frequenzschrittes. Der Schrittzähler wird bei jedem Wiederholungslauf zurückgesetzt und kann damit zur Steuerung der Ausgabe benutzt werden. Es gilt:

`nstep = 1`: Beginn einer neuen Simulation, die Ergebnisse des ersten Schrittes liegen vor;

`nstep > 1`: die Ergebnisse des `nstep`. Schrittes liegen vor;

`nstep = 0`: Ende der Simulation; es liegen keine Ergebnisse vor.

Die Variable `nstep` darf nicht verändert werden!

Folgende COMMON-Bereiche werden benötigt:

```
1       COMMON /CNTRLS/  CNTRL
2       REAL*8           CNTRL(160)
```

Der Vektor `CNTRL` enthält allgemeine Steuerinformationen:

`CNTRL(1)`	`(START) TIME` (TR-Analyse)
`CNTRL(2)`	`STOP TIME` (TR-Analyse)
`CNTRL(24)`	`NORQST`: Anzahl der Ausgabevariablen
`CNTRL(32)`	RUN/RERUN Nummer
`CNTRL(71)`	0 = TR-Analyse, 1 = AC-Analyse
`CNTRL(72)`	`INITIAL FREQUENCY` (AC-Analyse)
`CNTRL(73)`	`FINAL FREQUENCY` (AC-Analyse)
`CNTRL(74)`	`TYPE FREQUENCY RUN`, $\leq 1 =$ `LINEAR`, $> 1 =$ `LOG`
`CNTRL(75)`	`NUMBER FREQUENCY STEPS`
`CNTRL(20)`	`X PLOT DIMENSION`
`CNTRL(21)`	`Y PLOT DIMENSION`

Die Namen der Ausgabevariablen erhält man von

```
      COMMON /OUTPUT/ TPINDX
      CHARACTER*8     TPINDX (NORQST)
```

Für `nstep>0` stehen bei jedem Unterprogrammaufruf `NORQST` Werte zur Verfügung, und zwar in

```
      COMMON /OUTDTA/ OUTBFR
      REAL*8          OUTBFR (NORQST)
      COMPLEX*16      OUTPAC (NORQST)
      EQUIVALENCE    (OUTPAC(1),OUTBFR(1))
```

Der Name der unabhängigen Variablen `TIME/FREQ` steht unter `TPINDX(1)`, die aktuellen Werte in `OUTBFR(1)` bzw. `REAL(OUTPAC(1))`.

Externe Dateien müssen über den COMMON-Block `/TAPES/` und den installationsspezifischen Unterprogrammen `QQQOPN` und `QQQCLS` verwaltet werden.

Die Maximal- und Minimalwerte der unabhängigen Variablen `TIME` bzw. `FREQ` erhält man aus den Kontrollvariablen `CNTRL(1), CNTRL(2)` bzw. `CNTRL(72), CNTRL(73)`. Entsprechende Werte für die abhängigen Variablen liegen vor der Simulation nicht vor.

7.3.2 Beispiel zur Ausgabe mittels ANSI Treibers

Das folgende Programmbeispiel zeigt die Ausgabe der Ergebnisse mittels direkter Cursorsteuerung durch eine ANSI Escape-Zeichenfolge. Die Routine berücksichtigt die unterschiedlichen Formate bei der TR- und AC-Analyse. Ausgeführt wird diese „Digitalanzeige“ nur dann, wenn die Kontrollvariable `cntrl(20) = X PLOT DIMENSION` ungleich 0 ist.

```
      subroutine intplt (nstep)

      COMMON /CNTRLS/  CNTRL
         REAL*8        CNTRL(160)
      COMMON /OUTPUT/  TPINDX
         CHARACTER*8   TPINDX (2)
      COMMON /OUTDTA/  OUTBFR
         REAL*8        OUTBFR (2)
         COMPLEX*16    OUTPAC (2)
         EQUIVALENCE  (OUTPAC(1),OUTBFR(1))
      character        wpos*20,wclear*4
      character*12     wnumbr
      logical          lfirst,lac
      data  lfirst     /.true./

      if (nint(cntrl(20)).eq.0) go to 999

      if (lfirst) then
         lfirst=.false.
         norqst=nint(cntrl(24))
         lac=nint(cntrl(71)).ne.0
         wclear(1:1)=char(27)
         wclear(2:4)='[2J'
         wpos(1:2)=wclear(1:2)
         print *, wclear
      end if
```

```
27
28       if (nstep.eq.0) then
29          pause
30          print *, wclear
31          go to 999
32       end if
33
34       if (nstep.eq.1) then
35          do 10 i=1,norqst
36             write (wpos(3:),'(i2.2,'';1H'',a)') i+3,tpindx(i)
37             print *, wpos
38    10    continue
39       end if
40
41 C nstep .ge. 1:
42
43       do 20 i=1,norqst
44          if (lac) then
45             write (wnumbr,'(1p,e12.4)') aimag(outpac(i))
46             write(wpos(3:),'(i2.2,'';21H'',a)') i+3,wnumbr
47             print *, wpos
48             write (wnumbr,'(1p,e12.4)') real(outpac(i))
49             write (wpos(3:),'(i2.2,'';9H'',a)') i+3,wnumbr
50           else
51             write (wnumbr,'(1p,e12.4)') outbfr(i)
52             write (wpos(3:),'(i2.2,'';9H'',a)') i+3,wnumbr
53          end if
54          print *, wpos
55    20 continue
56
57   999 continue
58       return
59       end
```

7.3.3 Das Unterprogramm POST

Das Unterprogramm POST wird *nach jedem* Simulations-/ Wiederholungslauf aufgerufen und wertet eine interne, unformatierte Datei INOUT aus. In ihr sind *alle* berechneten Ergebnisse enthalten. Standardmäßig kopiert POST diese Ergebnisse auf eine direkte Zugriffsdatei dirac (siehe Kapitel 7.4). Will man eigene Grafikprogramme an dieser Stelle ausführen, muss dieses Unterprogramm durch eine Routine gleichen Namens ersetzt werden, welches folgende Struktur aufweisen muss:

```
1        SUBROUTINE POST (lpost)
2
3        COMMON /TAPES/    NOUTP,INTAPE,LIBTP,INOUTP
4        COMMON /CNTRLS/   CNTRL(160)
5        DOUBLE PRECISION CNTRL,XRUNNO
6           EQUIVALENCE   (CNTRL(32),XRUNNO)
7
8        CHARACTER*8       HEADNG(9,11)
9        CHARACTER*8       OUTREQ(100)
10       DOUBLE PRECISION TEMP,XTYPE
11       REAL              BUFFER(3000)
12          COMPLEX        CBUF(1500)
13          EQUIVALENCE   (CBUF(1),BUFFER(1))
```

```
      CHARACTER*1       CK(100)
      LOGICAL           LAC,lpost

      REWIND (INOUTP)
C nrunno=0: master run, > 0: rerun number
      NRUNNO=NINT(XRUNNO)-1
 C AC or TR run ?
      READ (INOUTP) XTYPE
      LAC=.FALSE.
      IF (NINT(XTYPE).EQ.1) LAC=.TRUE.
C headings:
      READ (INOUTP) TEMP
      LENHDG=NINT(TEMP)
      IF (LENHDG.GT.0) THEN
         DO 10 N=1,LENHDG
            READ (INOUTP) (HEADNG(M,N),M=1,9)
   10    CONTINUE
      END IF
C no. of output requests:
      READ (INOUTP) TEMP
      LNINDX=NINT(TEMP)
      DO 20 N=1,LNINDX
         READ (INOUTP) OUTREQ(N)
   20 CONTINUE
C AC formats (i.e. DEGEGREES, RADIANS, COMPLEX) :
      IF (LAC) READ (INOUTP) (CK(I),I=1,LNINDX)
      KOUNT=0

C fill output buffer:
   30 READ (INOUTP) NGRPS
      IF (NGRPS.EQ.0) GO TO  99
      KOUNT=KOUNT+NGRPS
      LEND=NGRPS*LNINDX
      IF (LAC) THEN
         READ (INOUTP) (CBUF(K),K=1,LEND)
       ELSE
         READ (INOUTP) (BUFFER(K),K=1,LEND)
      END IF
C CBUF/BUFFER now contains the results of the LNINDX output vaiables
C for NGRPS integration steps (NGRPS*LNINDX)
      GO TO 30

  999 CONTINUE
      WRITE (NOUTP,608) KOUNT, NRUNNO
  608 FORMAT ('0POST - NO. OF SOLUTION POINTS: ',I6,10X,' RUN: ',I3)
      REWIND (INOUTP)
 9999 continue
      lpost=.false.
      RETURN

      END
```

Man beachte, dass bei Wiederholungsläufen die Datei `INOUT` mit den neuen Ergebnissen überschrieben wird. Sollen die Ergebnisse von verschiedenen Simulationsläufen in einer Grafik miteinander verglichen werden, muss der Benutzer entsprechende Vorkehrungen treffen.

Alle Ergebnisse werden mit einfacher Genauigkeit abgespeichert.

Bei der AC-Analyse enthält die Datei `INOUTP` zusätzlich den Vektor `CK`. Pro Ausgabevariable beschreibt ein Zeichen die Art der AC Ergebnisse:

`CK(i)`	`REAL(CBUF(i))`	`AIMAG(CBUF(i))`
`C`	Realteil	Imaginärteil
`R`	Amplitude	Winkel in (rad)
`D`	Amplitude	Winkel in (°)

7.4 Auswertung der Ergebnisse ("Postprocessing")

Standardmäßig erzeugt SCEPTRE nach jeder Simulation eine direkte Zugriffsdatei (DA) `dirac`, die alle Ergebnisvektoren der TR- und AC-Analyse enthält, die der Benutzer unter der Eingabeüberschrift `OUTPUTS` angefordert hat. Der Inhalt dieser DA-Datei wird im Ausdruck `liste2` protokolliert (Abb. 7.2). Die Namen der Ausgabevariablen in der direkten Zugriffsdatei enthalten zusätzlich zwei Informationen:

```
directory of direct access file

no. symbol_name type length address   last access

     1   DIRECTOR   0      0       1 01/06/92 18:50:48
     2   HEADING    9     72       2 01/06/92 18:50:48
     3   TIME  _0   4   3038       3 01/06/92 18:50:49
     4   VCE   _0   4   3038       9 01/06/92 18:50:50
     5   VCC   _0   4   3038      15 01/06/92 18:50:52
     6   VC1   _0   4   3038      21 01/06/92 18:50:53
     7   IR3   _0   4   3038      27 01/06/92 18:50:54
     8   J1    _0   4   3038      33 01/06/92 18:50:55
     9   TIME  _1   4   2870      39 01/06/92 18:51:14
    10   VCE   _1   4   2870      45 01/06/92 18:51:15
    11   VCC   _1   4   2870      51 01/06/92 18:51:16
    12   VC1   _1   4   2870      57 01/06/92 18:51:17
    13   IR3   _1   4   2870      63 01/06/92 18:51:19
    14   J1    _1   4   2870      69 01/06/92 18:51:20
    15   TIME  _2   4   2962      75 01/06/92 18:51:39
    16   VCE   _2   4   2962      81 01/06/92 18:51:40
    17   VCC   _2   4   2962      87 01/06/92 18:51:41
    18   VC1   _2   4   2962      93 01/06/92 18:51:42
    19   IR3   _2   4   2962      99 01/06/92 18:51:44
    20   J1    _2   4   2962     105 01/06/92 18:51:45
    21   ZZZZZZZZ   0      0     111 01/06/92 18:51:45
```

Abbildung 7.2: Protokoll zur DA-Datei

den Typ der Ausgabevariablen und die Nummer des Wiederholungslaufes, aus dem diese Variable stammt. Der Name ist wie folgt aufgebaut:

variable *t* *n*

n enthält die Laufnummer (0 entspricht dem *master run*), das Zeichen *t* beschreibt den Typ der Ausgabevariablen entsprechend folgender Tabelle:

t	Typ der Ausgabevariablen
_	Variable stammt aus einer TR-Analyse
I	AC Analyse, Imaginärteil
D	AC Analyse, Phasenwinkel in Grad
R	AC Analyse, Phasenwinkel im Bogenmaß
A	AC Analyse, Amplitude bzw. Realteil

Am einfachsten lassen sich die Ergebnisse grafisch auswerten mit dem Programm NGP (siehe 7.5). Für eigene Anwendungen bestehen die Möglichkeiten, entweder die internen Routinen `INTPLT` und `POST` durch eigene Unterprogramme zu ersetzen (siehe Kapitel 7.3) oder die DA-Datei selber zu verarbeiten (siehe [18]).

7.5 Das Grafikprogramm NGP

Das Programm `NGP` dient zur grafischen Darstellung von Vektordaten aus der direkten Zugriffsdatei (DA) `dirac`. Es können maximal 6 Kurven in ein Diagramm gezeichnet werden. Die Auswahl der zu zeichnenden Ergebnisvektoren geschieht mittels Indizes statt symbolischer Namen. Die Darstellung der Kurven erfolgt in einem selbstskalierenden Koordinatensystem. Werden mehrere Kurven gleichzeitig dargestellt, richtet sich die Skalierung automatisch nach den Maxima und Minima aller Kurven. Enthält die DA-Datei Ergebnisvektoren aus Wiederholungsläufen (siehe 7.4), können diese Ergebnisse ebenfalls gleichzeitig dargestellt werden. Das Programm zeichnet automatisch jede y-Variable mit der dazugehörenden x-Variablen.

7.5.1 Übersicht der Kommandos

Zu Beginn und nach jedem NGP-Kommando werden alle Vektoren mit ihrem Index, Minimum, Maximum, Erstellungsdatum und -uhrzeit aufgelistet. Der Index wird bei den Kommandos `X`, `Y` und `S` benötigt.

```
X n0
```

Als x-Werte (Abszisse) werden die Werte der `n0`. Variable genommen. Es kann nur eine Variable angegeben werden. Bei der gleichzeitigen Darstellung von Kurven aus Wiederholungsläufen wird automatisch für `n0` der passende Index für die dazugehörigen X-Werte eingesetzt.

```
Y n1 n2 ...
```

In einer Zeichnung werden gleichzeitig die Werte der `n1.`,`n2.`…Variable als y-Werte (Ordinate) gezeichnet.

```
S n0 n1 n2 ...
```

Die `n0.` Variable wird als x-Wert, die `n1., n2.`…werden als y-Werte benutzt. Dies ist die schnellste Art, eine Zeichnung zu generieren. Für `n0` gilt das gleiche wie beim `X`-Befehl.

```
L
```

Listet die aktuellen Parameter auf.

`D` oder `P`

Erstellt eine vollständige Zeichnung entsprechend der aktuellen Parameter.

`E` oder `Q`

Beendet das Programm.

Alle darzustellenden Größen können auf den Bereich ± 100 % normiert werden mit

`scale on` und `scale off`.

Der Skalierungsfaktor wird beim `L`-Befehl ausgegeben.

Hinweis: Bei der Darstellung von Diagrammen, die aus Wiederholungsläufen stammen, darf der Index (`n0`) für die Ordinate (x-Werte) nur einmal angegeben werden. Das Programm wählt automatisch die passenden X-Werte aus den Wiederholungsläufen aus.

Beispiel: aus der DA-Datei nach Abb. 7.2 sollen die Spannungen `VCC` und `VC1` von allen Rechenläufen gleichzeitig als Funktion von `TIME` in einer Zeichnung dargestellt werden:

```
s 3 5 6 11 12 17 18
```

Dies gilt auch für parametrische Kennlinien, bei denen die X-Werte nicht die Zeit darstellen, z.B. $V_{CC} = f(V_{C1})$:

```
s 6 5 11 17
```

7.5.2 NGP und GNUPLOT

Das NGP-Programm schreibt die zu zeichnenden Vektoren in ASCII-Dateien und erzeugt gleichzeitig eine Steuerdatei `temp.GPI`. Anschließend wird aus NGP_GNU heraus[1] das Grafikpaket **gnuplot** mit dem Kommando `gnuplot temp.GNU` gestartet und damit die ausgewählten Vektoren automatisch gezeichnet.

Diese Version läuft nur unter UNIX-ähnlichen Betriebssystemen.

Die Dateinamen für die ASCII-Dateien, die die Wertepaare der zu zeichnenden Vektoren enthalten, werden nach folgendem Schema gebildet:

```
yyyyyy_n.xxxxxx.GPI
```

Es bedeuten:

`yyyyyy` SCEPTRE-Output-Variable für die Ordinate

`_n` Nummer des Laufes

`xxxxxx` SCEPTRE-Output-Variable für die Abszisse.

[1] mit dem Unterprogramm `SYSTEM`

Zur Auswahl des Ausgabegerätes gibt es einen neuen Befehl:

```
TERM 'cmd'
```

Für `cmd` muss ein gültiger GNUPLOT-Befehl angegeben werden, z. B. `'set term X11'` für das X Window System, `'set term kc_tek40XX'` für die farbige Ausgabe mittels der Kermit-Emulation (MS-DOS). Dieses Kommando überschreibt die Befehle, die in der Konfigurationsdatei `ngp.rc` (s. u.) unter der Kennung `term:` eingetragen sind.

Eine Papierkopie (*hardcopy*) zu der zuvor auf dem Bildschirm dargestellten Grafik läßt sich mit dem Befehl

```
hc
```

erstellen. Es werden die GNUPLOT-Befehle ausgeführt, die in der Konfigurationsdatei `ngp.rc` (s. u.) unter der Kennung `hc:` eingetragen sind.

Zusätzliche GNUPLOT-Befehle, die nur temporär wirken, können mit dem Kommando

```
cmd 'kommandos'
```

ausgeführt werden. Die unter `'kommandos'` angegebenen Befehle, evtl. durch `;` getrennt, überschreiben die in der Konfigurationsdatei `ngp.rc` unter der Kennung `cmd:` eingetragenen Befehle.

7.5.3 Konfigurationsdatei `ngp.rc`

Beim Aufruf von NGP wird als Terminal und für die Hardcopy-Ausgabe der Terminaltyp `dumb` (ASCII-Terminal) angenommen. Die Handhabung von NGP läßt sich durch eine Konfigurationsdatei vereinfachen. Diese Datei enthält GNUPLOT-Kommandos zur Voreinstellung und muss im Verzeichnis `$HOME/etc` unter dem Namen `ngp.rc` angelegt werden. Die Datei enthält eine Kennung und anschließen einen oder mehrere GNUPLOT-Befehle (durch `;` getrennt). Kennung und Befehle dürfen 72 Zeichen nicht überschreiten. Die Reihenfolge der Kennungen ist beliebig. Folgende Kennungen sind vorgesehen:

`term:` setzt den Terminaltyp und evtl. die Ausgabedatei

`hc:` setzt den Terminaltyp und die Ausgabedatei für die Hardcopy-Ausgabe

`cmd:` GNUPLOT Befehle, die bei jeder Zeichnung ausgeführt werden.

Beispiel:

```
hc: set term post land; set out "|lpr"
term: set term X11
cmd: set grid; set time
```

Für die Papierkopie wird PostScript im Querformat ausgewählt und die erzeugte Datei direkt an das Druckkommando weitergegeben. Die Grafik wird auf einem X11-Terminal ausgegeben. Jede Grafik wird mit Gitterlinien und Datum/Uhrzeit versehen.

7.6 Fortsetzungsläufe (CONTINUE)

Es ist möglich, eine TR-Analyse fortzusetzen (siehe Abschnitt 2.14, Seite 36). In diesem Fall besteht die SCEPTRE-Eingabedatei nur aus folgenden Anweisungen:

```
CONTINUE
RUN CONTROLS
   STOP TIME = ....
END
```

Für …ist die neue Simulationszeit einzusetzen, die größer sein muss als die des vorangegangen Laufes. Gestartet wird der Fortsetzungslauf, in dem man SCEPTRE mit dieser Datei aufruft. Weiterhin ist darauf zu achten, dass die Dateien `pgmsave.for` und `datasave.dat` von dem fortzusetzenden Lauf vorhanden sein müssen. Da diese Dateien durch jeden SCEPTRE-Lauf überschrieben werden, muss der Fortsetzungslauf unmittelbar der fortzusetzenden TR-Analyse folgen oder man kopiert/reserviert die beiden Dateien nach dem fortzusetzenden Lauf.

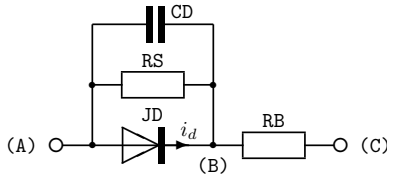
Teil IV

Anhang

A Dioden- und Transistor-Ersatzschaltbilder

A.1 Diode

Zur Simulation von Halbleiterelementen gibt es verschiedene Modelle. In den nachfolgenden Abbildungen A.1, A.2 und A.3 werden die Ersatzschaltungen einer Diode und bipolaren pnp/npn-Transistoren sowie ihre Darstellung in SCEPTRE beschrieben. Die Ersatzschaltungen beruhen auf dem Ebers-Moll Modell. Die in den SCEPTRE-Anweisungen kursiv gedruckten Parameter müssen aus den Datenblätter der entsprechenden Halbleiter entnommen werden. Dabei ist auf ein konsistentes Maßeinheitensystem zu achten.

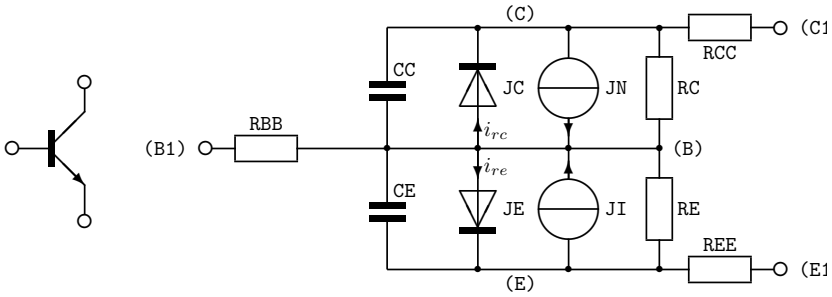


Parameter: $i_s, \Theta, R_B, R_S, C_0, V_0, n, K$	`MODEL` *name* `(A-C)` `ELEMENTS`

$i_d = i_s(e^{\Theta V_{JD}} - 1)$
$C_D = \frac{C_0}{(V_0 - V_{JD})^n} + K(i_d + i_s)$
	`JD, A-B =` `DIODE EQUATION` (i_s, Θ)
	`CD, A-B = Q1 (` $C_0, V_0, V_{JD}, n, K, J_D, i_s$ `)`
	`RB, B-C =` R_B
	`RS, A-B =` R_S
	`FUNCTIONS`
	`Q1 (A,B,C,D,E,F,G)=(A/(B-C)**D+E*(F+G))`

Abbildung A.1: SCEPTRE Diodenersatzschaltbild

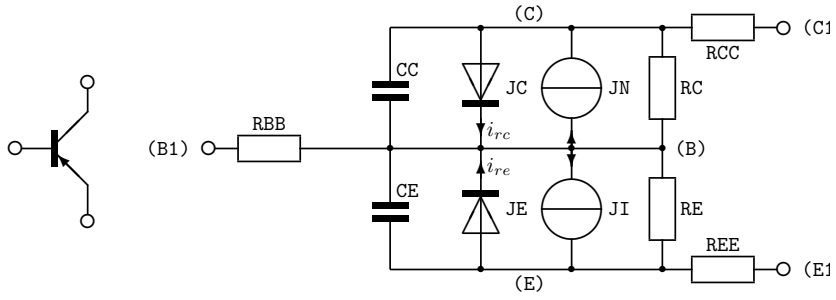
A.2 pnp Transistor



Parameter: $i_{es}, \Theta_e, \alpha_i, i_{cs}, \Theta_c, \alpha_n, C_{0e}, V_{0e}, K_e, n_e, C_{0c}, V_{0c}, K_c, n_c, R_{BB}, R_{EE}, zR_{CC}, R_E, R_C$	`MODEL` *name* `(B1-E1-C1)`
	`ELEMENTS`
$i_{re} = i_{es}(e^{\Theta_e V_{BE}} - 1)$	`JE, B-E =` DIODE EQUATION (i_{es}, Θ_e)
$i_{rc} = i_{cs}(e^{\Theta_c V_{BC}} - 1)$	`JC, B-C =` DIODE EQUATION (i_{cs}, Θ_c)
$i_i = \alpha_i i_{rc}$	`JI, E-B =` α_i `* JC`
$i_n = \alpha_n i_{re}$	`JN, C-B =` α_n `* JE`
$C_e = \frac{C_{0e}}{(V_{0e} - V_{BE})^{n_e}} + K_e(i_{re} + i_{es})$	`CE, B-E = Q1 (` $C_{0e}, V_{0e}, V_{CE}, n_e, K_e, J_E, i_{es}$)
$C_c = \frac{C_{0c}}{(V_{0c} - V_{BC})^{n_c}} + K_c(i_{rc} + i_{cs})$	`CC, B-C = Q1 (` $C_{0c}, V_{0c}, V_{CC}, n_c, K_c, J_C, i_{cs}$)
	`RBB, B1-B =` R_{BB}
	`REE, E1-E =` R_{EE}
	`RCC, C1-C =` R_{CC}
	`RE, E-B =` R_E
	`RC, C-B =` R_C
	`FUNCTIONS`
	`Q1 (A,B,C,D,E,F,G)=(A/(B-C)**D+E*(F+G))`

Abbildung A.2: SCEPTRE npn Transistor-Ersatzschaltbild

A.3 npn Transistor



Parameter: $i_{es}, \Theta_e, \alpha_i, i_{cs}, \Theta_c, \alpha_n,$ $C_{0e}, V_{0e}, K_e, n_e, C_{0c}, V_{0c}, K_c, n_c,$ $R_{BB}, R_{EE}, R_{CC}, R_E, R_C$	`MODEL` *name* `(B1-E1-C1)` `ELEMENTS`
$i_{re} = i_{es}(e^{\Theta_e V_{BE}} - 1)$	`JE, E-B = DIODE EQUATION (` i_{es}, Θ_e `)`
$i_{rc} = i_{cs}(e^{\Theta_c V_{BC}} - 1)$	`JC, C-B = DIODE EQUATION (` i_{cs}, Θ_c `)`
$i_i = \alpha_i i_{rc}$	`JI, B-E =` α_i `* JC`
$i_n = \alpha_n i_{re}$	`JN, B-C =` α_n `* JE`
$C_e = \frac{C_{0e}}{(V_{0e} - V_{BE})^{n_e}} + K_e(i_{re} + i_{es})$	`CE, E-B = Q1 (` $C_{0e}, V_{0e}, V_{CE}, n_e, K_e, J_E, i_{es}$`)`
$C_c = \frac{C_{0c}}{(V_{0c} - V_{BC})^{n_c}} + K_c(i_{rc} + i_{cs})$	`CC, C-B = Q1 (` $C_{0c}, V_{0c}, V_{CC}, n_c, K_c, J_C, i_{cs}$`)`
	`RBB, B1-B =` R_{BB}
	`REE, E1-E =` R_{EE}
	`RCC, C1-C =` R_{CC}
	`RE, E-B =` R_E
	`RC, C-B =` R_C
	`FUNCTIONS`
	`Q1 (A,B,C,D,E,F,G)=(A/(B-C)**D+E*(F+G))`

Abbildung A.3: SCEPTRE pnp Transistor-Ersatzschaltbild

B Anhänge zu Vol. I [2]

B.1 Übersicht der SCEPTRE-Anweisungen

Tabelle B.1 enthält eine Übersicht aller SCEPTRE-Anweisungen und deren internen Variablen im COMMON-Block `CNTRLS`, die einen Bezug zu diesen Anweisungen haben. Der Index gilt für den Vektor `CTROLS` des selben COMMON-Blocks (siehe nächster Abschnitt B.2)

Tabelle B.1: SEPTRE-Anweisungen und die dazugehörigen Variablen im COMMON-Block `CNTRLS`

SCEPTRE-Anweisung	COMMON Variable	Index für `CTROLS`
START TIME	TIME	1
STOP TIME	XSTOPT	2
MAXIMUM STEP SIZE	XMXISS	7
INTEGRATION ROUTINE	XIR	4
STARTING STEP SIZE	XTISSS	5
MINIMUM STEP SIZE	XMNISS	6
MINIMUM ABSOLUTE ERROR	XMNAIE	8
MAXIMUM ABSOLUTE ERROR	XMXAIE	9
RUN INITIAL CONDITIONS	XIC	10
COMPUTER TIME LIMIT	XMXERT	3
RUN INITIAL CONDITIONS ONLY	XTRANS	34
MINIMUM RELATIVE ERROR	XMNRIE	13
MAXIMUM RELATIVE ERROR	XMXRIE	14
RELATIVE CONVERGENCE	XICRER	17
ABSOLUTE CONVERGENCE	XICAER	18
X PLOT DIMENSION	XXDIMS	20
Y PLOT DIMENSION	XYDIMS	21
MAXIMUM PRINT POINTS	XMXOPT	19
NO ELEMENT SORT		
MAXIMUM INTEGRATION PASSES	XMXPAS	12
NEWTON-RAPHSON PASS LIMIT	XMXICP	16
PUNCH BINARY CARDS		

Fortsetzung nächste Seite

Fortsetzung von Tabelle B.1

SCEPTRE-Anweisung	COMMON Variable	Index für CTROLS
FULIST		
PUNCH PROGRAM		
WRITE SIMUL8 DATA		
COMPUTER SAVE INTERVAL	XSAVE	35
VECTOR EQUATIONS		
WRITE DEBUG		
PRINT B MATRIX		
SOLUTION TIME LIMIT	XMXSLT	37
PLOT INTERVAL	XPLTI	38
PRINT INTERVAL	XPRTI	39
LIST NODE MAP		
IC FOR RERUNS	XSAVIC	53
RUN IC VIA IMPLICIT	XTRNIC	55
USE DIFFERENCED JACOBIAN	XDIFF	61
USE SYMBOLIC JACOBIAN	XSYMBL	62
EXECUTE SETUP PHASE ONLY		
RUN AC	XAC	71
INITIAL FREQUENCY	XINFRQ	72
FINAL FREQUENCY	XFNFRQ	73
TYPE FREQUENCY	XTYPFQ	74
NUMBER FREQUENCY STEPS	XNFRQS	75
PRINT A MATRIX		
PRINT EIGENVALUES		
PRINT EIGENVECTORS		
RUN MONTE CARLO	XNMCPS	121
RUN OPTIMIZATION	XNOPPS	131
RUN WORST CASE	XWCLHN	151
FREQUENCY	FREQ	79
INITIAL RANDOM NUMBER	XRANNO	122
DISTRIBUTION	XDISTR	125
LIST MONTE CARLO DETAILS		
USE FIXED AC MATRIX IN RERUNS	XACRE	85
IMPULSE RESPONSE BUFFER	XHAVE	87
INPUT FUNCTION BUFFER	XIAVE	88
COMPRESSION CRITERION	XCMPCR	89
COMPRESSION COUNT	XCMPCT	90
LIST OPTIMIZATION DETAILS		

Fortsetzung nächste Seite

Fortsetzung von Tabelle B.1

SCEPTRE-Anweisung	COMMON Variable	Index für CTROLS
PUNCH OPTIMIZATION RESULTS		
OPTIMIZATION CRITERION	XOCPR	134
OPTIMIZATION RANDOM STEPS	XNOPRS	135
RANDOM STEP SIZE CONTROL	XRNSSC	136
INITIAL H MATRIX FACTOR	XHMFAC	137
RUN SENSITIVITY	XSENST	141
MINIMUM FUNCTION ESTIMATE	XMNFES	138

B.2 Interne Variablen — COMMON-Bereich /CNTRLS/

Zugriff auf die internen Variablen erhält man über den COMMON-Bereich /CNTRLS/. Man kann die Variable sowohl mit ihrem Namen als auch über den Vektor CTROLS und dem entsprechenden Index aus Tabelle B.1 ansprechen.

```
 IMPLICIT REAL*8(A-J,L-M,O-Z),INTEGER*4(K,N)
 COMMON /CNTRLS/
*   TIME  ,XSTOPT,XMXERT,XIR   ,XTISSS,XMNISS,XMXISS,
*   XMNAIE,XMXAIE,XIC   ,XRERNO,XMXPAS,XMNRIE,XMXRIE,
*   XMNISO,XMXICP,XICRER,XICAER,XMXOTP,XXDIMS,XYDIMS,
*   XICPAS,XNOPRQ,XNTNDX,XNHEAD,XNDFEQ,XERT  ,XNRERN,
*   XCNTNU,XSTPNO,XPASNO,XRUNNO,XSTPSZ,XTRANS,XSAVE ,
*   XTMON ,XMXSLT,XPLTI ,XPRTI ,XMCPUT,SP1(3),XWRTSM,
*   SP2(8),XSAVIC,XGER  ,XTRNIC,SP3(5),XDIFF ,XSYMBL,
*   SP4(8),XAC   ,XINFRQ,XFNFRQ,XTYPFQ,XNFRQS,SP5(3),
*   FREQ  ,XNWDBM,SP6(3),XDERIV,XACRE ,XNCONV,XHAVE ,
*   XIAVE ,XCMPCR,XCMPCT,XNEY  ,XNE7DT,XNBC  ,XNBR  ,
*   XNBL  ,XNLC  ,XNLR  ,XNLL  ,XNJX  ,XNJ8DT,XNJ0  ,
*   XNJ9  ,XNJ8AC,XNE7AC,XNM33 ,XNM36 ,XNM66 ,XNDP  ,
*   XNDPD ,XNELTS,XLSW  ,XLSWP ,SP7(2),XICPRT,SP8(5),
*   XNMCPS,XRANNO(2),SP9,XDISTR,SPA(5),XNOPPS,SPB(2),
*   XOPCR ,XNOPRS,XRNSSC,XHMFAC,XMNFES,SPC(2),XSENST,
*   XADJNT,SPD(8),XWCLHN,SPE(9)
 DIMENSION CTROLS(160)
 EQUIVALENCE (CTROLS,TIME)
```

Die Werte der Kontrollvariablen dürfen nicht verändert werden.

B.3 Programmkapazität

Tabelle B.2 enthält die Programmgrenzen von SCEPTRE. Wird eine Grenze überschritten, wird der Lauf mit einer entsprechenden Fehlernachricht abgebrochen.

Tabelle B.2: Programmkapazität

Programmgröße	Maximum
Netzwerkelemente	300
Knotenpunkte	301
Ableitungen von Quellen `DE, DJ`	50
Definierte Parameter `P`	200
Definierte Parameter `W`	50
Definierte Parameter `DP`	200
Gegeninduktivitäten `M`	50
Argumente in `EQUATION`-Anweisungen	50
Tabellenänderungen in Modellen	15
`EQUATION`-Änderungen in Modellen	15
Ausgabeunterdrückungen in Modellen	10
Ausgabevariablen	100
Anfangsbedingungen	100
`EQUATION`-Anweisungen	insgesamt ca. 5000 Zeichen
Folgezeilen für `EQUATION`-Anweisung	20
Tabellen	insgesamt 800 Wertepaare
Abbruchbedingungen `TERMINATE, STOP`	10
Permanent gespeicherte Modelle	250
Externe Modellknotenpunkte	25
Interne Modellknotenpunkte	301
AC-Quellen 1. plus 2. Ordnung	50
Unabhängige AC-Strom- und Spannungsquellen	50
Kommentarkarten	11

C Anhänge zu Vol. II [3]

C.1 Schwach besetzte Matrizen (*Sparse-Matrix*-Technik)

Zur Lösung des Gleichungssystems

$$A\,X = b \tag{C.1}$$

mit A als Koeffizientenmatrix n. Ordnung, dem Vektor b, der die Anregungsfunktion enthält und den Unbekannten X, bedient man sich u. a. des Gaußschen Algorithmus. Der Rechenaufwand steigt etwa quadratisch mit der Ordnung n. Hinzu kommt noch, dass die Koeffizientenmatrix nur schwach besetzt ist, d. h. die Anzahl der mit Null besetzten Elementen steigt ebenfalls quadratisch. Bei großen Netzwerkproblemen ($n > 100$) fällt der Rechenaufwand derart ins Gewicht[1], dass man ein anderes Verfahren anwendet, die *LU-Faktorisierung*.

Falls A^{-1} existiert lässt sich A in zwei Dreiecksmatrizen L und U umwandeln:

$$A = L\,U \tag{C.2}$$

mit

$$L = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \qquad U = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & 1 & u_{23} & \cdots & u_{2n} \\ 0 & 0 & 1 & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Die Matrizen L und U werden aus der A-Matrix durch folgendes rekursives Verfahren gewonnen:

$$l_{iq} = a_{iq} - \sum_{k=1}^{q-1} l_{ik} u_{kq} \qquad \text{mit } l_{iq} = 0 \text{ for } i < q$$

$$u_{pj} = \frac{a_{pj} - \sum\limits_{k=1}^{p-1} l_{pk} u_{kj}}{l_{pp}} \qquad \text{mit } u_{pj} \begin{cases} = 0 \text{ für } p > j \\ = 1 \text{ für } p = j \end{cases}$$

Setzt man Gleichung (C.2) in Gleichung (C.1) ein, ergibt sich

$$L\,U\,X = b\,. \tag{C.3}$$

Mit dem neuen Vektor Y

$$Y = U\,X \tag{C.4}$$

[1] Diese Aussage stammt aus dem Jahr 1974 (Anm. des Autors).

ergibt sich für Gleichung (C.3)

$$L\,Y = b\,. \tag{C.5}$$

Gleichung (C.5) lässt sich durch Vorwärtsubstitution, Gleichung (C.4) mit der Lösung Y durch Rücksubstitution lösen und man erhält damit den Lösungsvektor X.

SCEPTRE aktiviert diese Technik nur bei großen Netzwerken mit mehr als 70 Elementen. Für kleinere Netzwerke wird die Gauss-Jordan Methode benutzt.

(siehe auch [6, Kap. 3.3])

C.2 Assemblerroutinen für die Ein-/Ausgabe

Ein Teil der FORTRAN Ein- und Ausgabeanweisungen, nämlich `WRITE, READ, ENDFILE` wurden durch die Assemblerroutinen `GET, PUT, ENDFIL` ersetzt, um Rechenzeit zu sparen. Damit wurde die Portabilität von SCEPTRE eingeschränkt. Die FORTRAN Ein-/Ausgabeanweisungen wurden im Quelltext auskommentiert. Für Bit- und Bytemanipulationen wurden ebenfalls Assemblerroutinen eingesetzt, und zwar `CPY, DAND, DOR, DEXOR, LBITS, XIBCD, SORT` und für das Datum `ITODAY`.

In der vorliegenden Version wurden alle Assemblerroutinen durch FORTRAN-77 Routinen ersetzt.

Tabelle C.1 zeigt die Verbesserung durch Assemblerroutinen bei den Rechenzeiten einiger Beispiele. Die Zeiten beziehen sich auf einen Rechner IBM S/360 Modell 65.

Tabelle C.1: Vergleich der Rechenzeiten für Phase I zwischen FORTRAN-I/O and Assembler-I/O

	CPU Zeit in (s)	
Test Schaltung	FORTRAN I/O	Assembler I/O
Beispiel 1, Kap. 4.1	11	5
Beispiel 2, Kap. 4.2	24	9
Beispiel 3, Kap. 4.3	23	6
Beispiel 4, Kap. 4.4	11	5
Schaltung mit 221 Elementen (nur TR-Analyse)	90	62

C.3 Das *implicite* Integrationsverfahren

Eine ausführliche Beschreibung (in englisch) findet man im Originalhandbuch Volume II [3, Anhang A].

C.4 Herleitung der B-Matrix

Die B-Matrix drückt die Spannungen in den Verbindungszweige (*link*) durch Spannungen in den Baumzweigen (*branch*) und Ströme in den Baumzweigen durch Ströme in den Verbindungszweigen aus.

Eine ausführliche Beschreibung (in englisch) findet man im Originalhandbuch Volume II [3, Anhang B].

C.5 Auswertung der Diodenkennlinie bei der Berechnung der Anfangsbedingungen

Siehe Originalhandbuch Volume II [3, Anhang C].

C.6 Die Newton-Raphson-Methode

Siehe Originalhandbuch Volume II [3, Anhang D].

C.7 Identität der Gleichungen (5.79) und (5.80)

Siehe Originalhandbuch Volume II [3, Anhang E].

C.8 Adjungiertes Netzwerk

Die Berechnung der Empfindlichkeit einer Schaltung (*sensitivity*) erfolgt mit Hilfe des adjungierten Netzwerks. Einzelheiten findet man im Originalhandbuch Volume II [3, Anhang F].

D Das Shell-Script `sceptre`

```
#!/bin/sh

BASEDIR=/usr/local
LIBX2=${BASEDIR}/lib/libx2.a
LIBX3=${BASEDIR}/lib/libx3.a
EXEC1=${BASEDIR}/bin/exec1
EXEC2=${BASEDIR}/lib/exec2.o
USRLIB=${BASEDIR}/lib/usrlib.a
MODELLIB=${BASEDIR}/lib/modellib.dat
FC='gfortran'
if test -f $HOME/lib/usrlib.a; then
  USRLIB=$HOME/lib/usrlib.a
fi
if test -f $HOME/lib/modellib.dat; then
  MODELLIB=$HOME/lib/modellib.dat
fi
if test -f $1.d
then
  cp $1.d input.dat
  ln -s ${MODELLIB} modellib.dat
  $EXEC1
  echo " press ENTER to continue ........"; read dummy
  if test -s sceptre.lis; then
    mv sceptre.lis $1.ls1
    asa < $1.ls1 | less
  fi
  echo " press ENTER to continue, CTRL-C to abort"; read dummy
  if test -s pgmsave.for
  then
    ${FC} ${FFLAGS} -o exec2 pgmsave.f $EXEC2 $USRLIB $LIBX2 $LIBX3 2>$1.log
    ./exec2
    if test -s sceptre.lis; then
      mv sceptre.lis $1.ls2
      asa < $1.ls2 | less
    fi
    if test -f dirac.dat
      then
      mv dirac.dat $1.dirac
    fi
    ls -l $1.*
    rm -f pgmsave.o exec2 input.dat # pgmsave.f acsave.dat datasave.dat
  else
```

```
43      echo 'simulation canceled'
44      exit
45    fi
46  else
47    echo 'file ' $1.d ' not found'
48  fi
49  exit
```

Literatur

[1] J.C. Bowers und S.R. Sedore. *SCEPTRE: A Computer Program for Circuit and System Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1971.

[2] D. Becker. *Extended SCEPTRE*. User's Manual. AFWL: TR-73-75, NTIS: ADA 009594.

[3] D. Becker. *Extended SCEPTRE*. Mathematical Formulation. AFWL: TR-73-75, NTIS: ADA 009595.

[4] R. W. Jensen und L. P. McNamee. *Handbook of Circuit Analysis Languages and Techniques*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1976.

[5] *Advanced Statistical Analysis Program (ASTAP)*. General Information Manual. IBM GH20-1271-0.

[6] H. Spiro. *Simulation integrierter Schaltungen durch universelle Rechnerprogramme. Verfahren und Praxis der rechnergestützten Simulation nichtlinearer Schaltungen*. München: Verlag Oldenbourg, 1985.

[7] W. Kaplan. *Ordinary Differential Equations*. Reading, Mass.: Addison Wesley, 1958. Kap. 10-2, S. 400–401.

[8] W. Kaplan. *Ordinary Differential Equations*. Reading, Mass.: Addison Wesley, 1958. Kap. 10-4, "Heun's Method", S. 402–403.

[9] K. G. Ashar, H. N. Ghosh, A. W. Aldridge und L. J. Patterson. "Transient Analysis and Device Characterization of ACP Circuits". In: *IBM Journal of Research and Development* 7 (1963), S. 218.

[10] P. E. Chase. "Stability Properties of Predictor-Corrector Methods for Ordinary Differential Equations". In: *Journal A.C.M* 9 (Okt. 1962), S. 457–468.

[11] Wm. C. Davidon. *Variable Metric Method for Minimization*. Report ANL-5990 (REV). AEC Research und Development, 1959.

[12] M. J. Box. "A Comparison of Several Current Optimization Methods, and the Use of Transformation in Constrained Problems". In: *The Computer Journal* 9 (1966), S. 67–77.

[13] M. J. Box, D. Davies und W. G. Svann. *Non-Linear Optimization Techniques*. Monograph No. 5. Imperial Chemical Industries Ltd., 1969.

[14] *Variable Metric Minimization*. SHARE Routine No. 1117, AN Z013, A3. Suite 750, 25 Broadway New York, NY 10004: SHARE Inc.

[15] W. Tinney und J. Walker. *Direct Solution of Sparse Network Equations by Optimally Ordered Triangular Factorization*. IEEE, Nov. 1967.

[16] J. Grad und M. A. Brebner. "Algorithm 343, Eigenvalues and Eigenvectors of a general matrix". In: *Comm. ACM* 11 (Dez. 1968), S. 820–826.

[17] J. V. Leeds und G. I. Ugron. "Simplified Multiple Parameter Sensitivity Calculation and Continuously Equivalent Network". In: *IEEE Trans. on Circuit Theory* 14 (Juni 1967).

[18] W.-R. Novender. "POS3 – ein Programmsystem zum symbolischen Verwalten von Vektoren auf einer direkten Zugriffsdatei". 2014.

[19] J. Portasik und C. Glass. *System Analysis Using the SCEPTRE Computer Program*. 98. AFWL, Sep. 1968.

[20] S. R. Sedore. "SCEPTRE: A Program for Automatic Network Analysis". In: *IBM Journal of Research and Development* 2 (Nov. 1967).

[21] S. R. Sedore. "More Efficient Use of the F Matrix in Practical Circuit Analysis Program". In: *IEEE Trans. on Computers* 17 (Mai 1968).

[22] M. Fowler und R. Warten. "A Numerical Integration Technique for Ordinary Differential Equations with Widely Separated Eigenvalues". In: *IBM Journal of Research and Development* 2 (Sep. 1967).

[23] R. H. Dickhaut. *Advances in Computer Techniques for Radiation Effects Calculation*. D2-90571. Boeing, 1964.

[24] W. C. Cordwell. *Radiation Device Modeling for the SCEPTRE Program*. 69-825-2371. Owego, N.Y.: IBM, Juni 1969.

[25] M. B. Reed und S. Seshu. *Linear Graphs and Electrical Networks*. Reading, Mass.: Addison-Wesley, 1961.

[26] F. B. Hildebrand. *Introduction to Numerical Analysis*. New York: McGraw-Hill, 1956.

[27] R. M. Warten. "Automatic Step-Size Control for Runge-Kutta Integration". In: *IBM Journal of Research and Development* 7.4 (Okt. 1963), S. 340–341.

[28] C. W. Gear. "The Numerical Integration of Ordinary Differential Equations". In: *Math. Comp.* 21 (1967).

[29] C. W. Gear. "The Automatic Integration of Ordinary Differential Equations". In: *Comm. of A.C.M.* 14 (März 1971).

[30] S. W. Director und R. A. Rohrer. "The Generalized Adjoint Network and Network Sensitivities". In: *IEEE Transaction on Circuit Theory* 16 (Aug. 1969).

[31] S. W. Director und R. A. Rohrer. "Automated Network Design – The Frequency-Domain Case". In: *IEEE Transaction on Circuit Theory* 16 (Aug. 1969).

AFWL: Air Force Weapons Laboratory, Kirtland, AFB, NM 87117, USA
NTIS: National Technical Information Service, Springfield, VA 22151, USA
AFAPL: Air Force Aero Propulsion Laboratory, Wright-Patterson AFB, Ohio 45433, USA

Index

*Hess*sche Matrix, 110
computational delay, 90

Baumzweige, 84

C-BAum, 84

Davidon-Verfahren, 110

Ebers-Moll ESB, 143
Eingabeformat, 11

Funktion
    `FPULSE`, 52
    `FSINSQ`, 53

implizites Euler-Verfahren, 99

Kippspannung, 51
Klassifizierung
    Elemente, 9
    SCEPTRE Variablen, 10

Lange
    Variablennamen, 9
LU-Faktorisierung, 151

Maßeinheiten, 10
Mehrschrittverfahren, 99
Modal Matrix, 113

Netzwerkbaum, 84
Nyquist Plot, 18

periodische Funktionen, 50

Rampenfunktion, 51
Rechteckfunktion, 51

Schnittmenge (*cut set*), 84

Verbindungszweig, 84

Zeitmultiplex-Signal, 51
Zustandsvariable, 86
Zählpfeile
    Quellen, 10