

RADC-TR-79-245, Vol II (of three)  
In-House Report  
September 1979



AD A 076596

# NONLINEAR CIRCUIT ANALYSIS PROGRAM (NCAP) DOCUMENTATION User's Manual

Mr. Jon B. Valente, RADC  
Ms. Sharon Stratakos, Syracuse University



A076317

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DDC FILE COPY

**ROME AIR DEVELOPMENT CENTER**  
**Air Force Systems Command**  
**Griffiss Air Force Base, New York 13441**

79 11 07 072

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

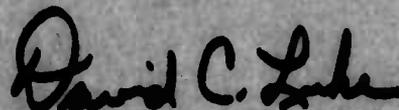
RADC-TR-79-245, Volume II (of three) has been reviewed and is approved for publication.

APPROVED:



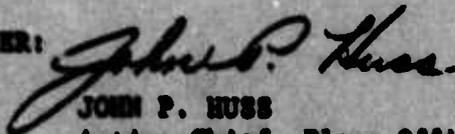
SAMUEL D. ZACCARI  
Chief, Compatibility Branch  
Reliability and Compatibility Division

APPROVED:



DAVID C. LUKE, Lt Colonel, USAF  
Chief, Reliability and Compatibility Division

FOR THE COMMANDER:



JOHN P. HUSS  
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBCT), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER 18 RADC-TR-79-245 Vol II (of three)	2. GOVT ACCESSION NO.	3. RECIPIENT CATALOG NUMBER 9	
4. TITLE (and Subtitle) 6 NONLINEAR CIRCUIT ANALYSIS PROGRAM (NCAP) DOCUMENTATION, Volume II. User's Manual.		5. TYPE OF REPORT & PERIOD COVERED In-house and Phase Report May 1978 - July 1979	
7. AUTHOR(s) 10 Jon B. Valente Sharon/Stratakos	8. CONTRACT OR GRANT NUMBER(s) 15 In-house and Contract F30602-79-C-0011		
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rome Air Development Center (RBCT) Griffiss AFB NY 13441 and Syracuse University, Syracuse NY 13210	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 23380314/23380317		
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RBCT) Griffiss AFB NY 13441	11. REPORT DATE September 1979		
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	13. SECURITY CLASS. (of this report) UNCLASSIFIED	12. NUMBER OF PAGES 74	13a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.	16 2338	17 03	12 82
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same			
18. SUPPLEMENTARY NOTES RADC Project Engineer: Jon B. Valente (RBCT)			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Electromagnetic Compatibility      Computer Program Nonlinear Analysis                      Volterra Analysis Nonlinear Circuit Analysis              Computer-aided Circuit Analysis			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Nonlinear Circuit Analysis Program (NCAP) is a circuit analysis code which uses the Volterra approach to solve for the transfer functions and node voltages of nonlinear circuits. To increase the transportability, the code was written in ANSI FORTRAN. The code was revised and documented in a joint effort using both in-house and contracted manpower. This documentation is a result of that effort. The documentation is made up of three volumes: the Engineering Manual, the User's Manual, and the Programmer's Manual. → next page (Cont'd)			

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

The Engineering Manual, Volume I, contains the introduction to the documentation, a description of the theory upon which NCAP is based, and the procedure for obtaining the input parameter data.

The User's Manual, Volume II, contains a detailed description of the NCAP input language. The description includes the input commands and data requirements. Examples are used throughout the manual to aid the user in understanding the input language.

The Programmer's Manual, Volume III, contains a subroutine by subroutine description of the code. Each subroutine has a trace map, functional flow diagram and narrative which will help the programmer to understand the structure of NCAP.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Dist _____	
App'd _____	
Dist	Available for special
<b>A</b>	

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

## Table of Contents

<b>SECTION 1</b>	<b>OPERATIONAL INTRODUCTION TO NCAP</b>	<b>2-3</b>
1.1	Design Philosophy	2-3
1.2	Circuit Elements	2-4
1.2.1	Linear and Nonlinear Elements	2-4
1.2.2	Circuit Excitation and Order of Analysis	2-8
1.3	Preliminary Data Preparation	2-9
1.4	Optional Features of NCAP	2-10
<b>SECTION 2</b>	<b>STRUCTURE OF THE NCAP INPUT LANGUAGE</b>	<b>2-12</b>
2.1	Control Statements	2-12
2.2	Data Statements	2-13
2.3	Input Format	2-13
2.4	Data Types	2-14
2.5	Input Error Detection	2-15
<b>SECTION 3</b>	<b>DETAILED DESCRIPTION OF NCAP STATEMENTS</b>	<b>2-17</b>
3.1	Comment	2-17
3.2	Segmentation	2-18
3.3	Sources	2-19
3.3.1	Independent Voltage Source	2-19
3.3.2	Linear Dependent Sources	2-27
3.3.3	Nonlinear Dependent Sources	2-30
3.4	Components	2-33
3.4.1	Linear Components	2-33
3.4.2	Nonlinear Components	2-37

3.5	Nonlinear Elements	2-38
3.5.1	Vacuum Diode	2-39
3.5.2	Vacuum Triode	2-40
3.5.3	Vacuum Pentode	2-42
3.5.4	Semiconductor Diode	2-44
3.5.5	Bipolar Junction Transistor	2-45
3.5.6	Junction Field Effect Transistor	2-47
3.6	Solution Modification	2-51
3.6.1	Linear Component Modification	2-52
3.6.2	Nonlinear Component Modification	2-55
3.6.3	Frequency Modification	2-56
3.6.4	Dependent Source Modification	2-62
3.6.5	Nonlinear Element Modification	2-63
3.7	Output Control	2-64
3.7.1	Suppressing Output of Circuit Element Data	2-65
3.7.2	Suppressing Output of Nodes and Orders	2-67
3.7.3	Plotted Output	2-68

**SECTION 1**

**OPERATIONAL INTRODUCTION TO NCAP**

## SECTION 1: OPERATIONAL INTRODUCTION TO NCAP

### 1.1 Design Philosophy

RADC NCAP is a nonlinear circuit analysis program, implemented on the Honeywell 6180 computer facility, which employs the Volterra analysis technique to compute the nonlinear transfer functions of electronic circuits. From the point of view of the user the NCAP system is composed of three elements: an input language, by which the user describes the circuit to be analyzed, a computational phase which solves the network problem on a nodal basis, and an output phase which prints and/or plots the desired results.

A primary consideration in the design of the NCAP software was that it be easy to learn and use. An in-depth knowledge of the analytical techniques employed is not essential for its efficient use nor is the circuit analyst expected to understand the complex programming routines which make up the system. The user is only required to be able to translate his circuit analysis problem into the appropriate NCAP input language statements. By means of this "language" he describes the circuit to be analyzed, the frequencies and order of analysis and the desired output. In turn, the system interprets these input statements, performs the nonlinear analysis and outputs the results in printed or plotted form.

## 1.2 Circuit Elements

NCAP uses a set of standard electronic circuit element models, and can analyze networks made up of interconnections of these elements. The Honeywell 6180 version can handle circuits of up to approximately 500 nodes.

### 1.2.1 Linear and Nonlinear Elements

The following circuit elements have been included in the NCAP system:

- Independent Voltage Source
- Linear Dependent Sources
- Nonlinear Dependent Sources
- Linear Components
- Nonlinear Components
- Vacuum Diode
- Vacuum Triode
- Vacuum Pentode
- Semiconductor Diode
- Bipolar Junction Transistor
- Field Effect Transistor

These circuit elements are depicted in Table 1. Device nodes are shown as xx, where xx represents a user-assigned node number. For internodal devices, both the node numbers represented by xx and yy are assigned by the user. However, the assignment of nodes for multi-node devices is performed automatically by the system, according to specific conventions built into the device model. In these instances, the node number xx is assigned by the user while the nodes represented by xx + 1, xx + 2, etc. are assigned by the system.

These circuit element depictions are presented here for

LINEAR COMPONENT



RESISTOR

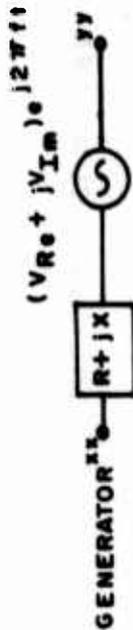


CAPACITOR



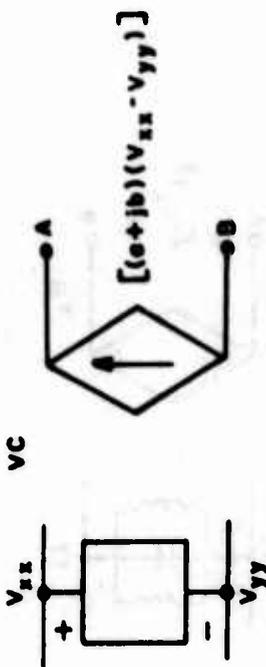
INDUCTOR

INDEPENDENT SOURCE

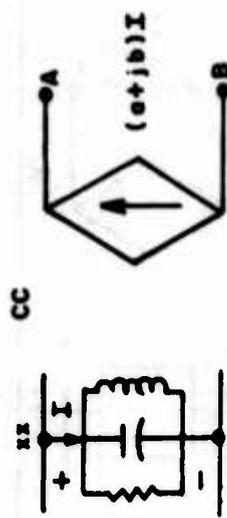


FREQUENCY =  $f$  Hz  
 AMPLITUDE =  $V_{Re} + jV_{Im}$   
 IMPEDANCE =  $R + jX$

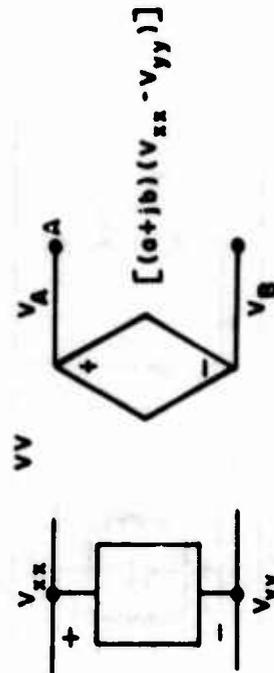
LINEAR DEPENDENT SOURCE



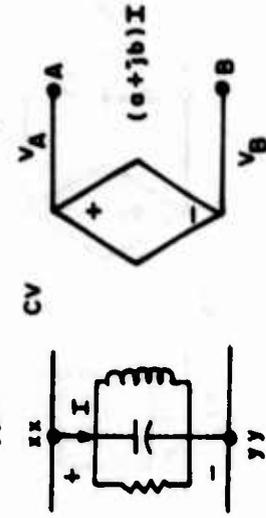
VC



CC



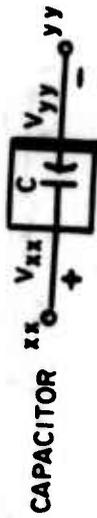
VV



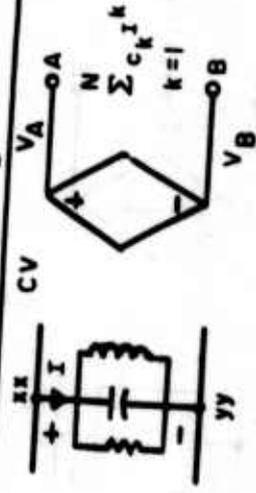
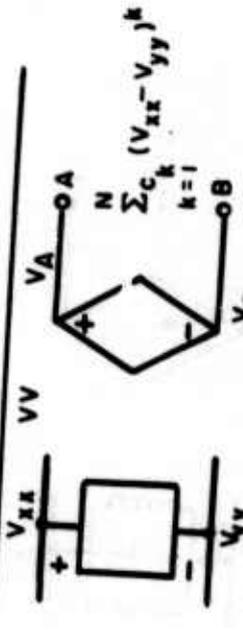
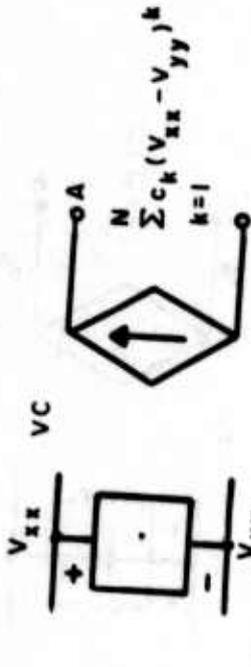
CV

LINEAR MODELING ELEMENTS

NONLINEAR COMPONENT

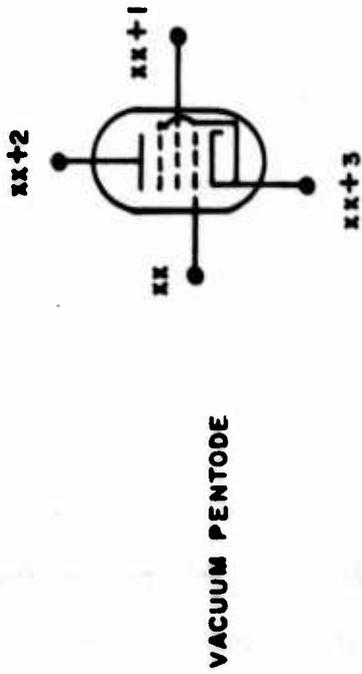
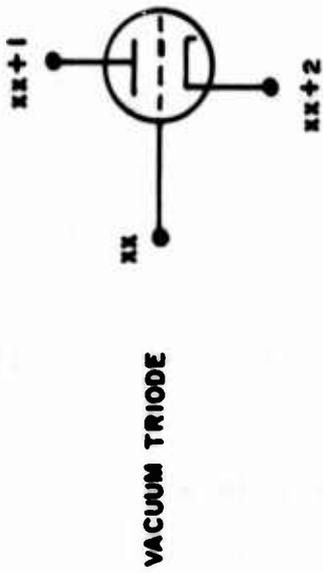


NONLINEAR DEPENDENT SOURCE

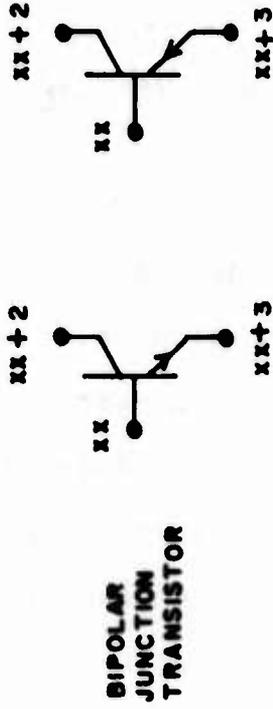
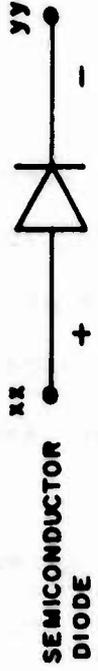


NONLINEAR MODELING ELEMENTS

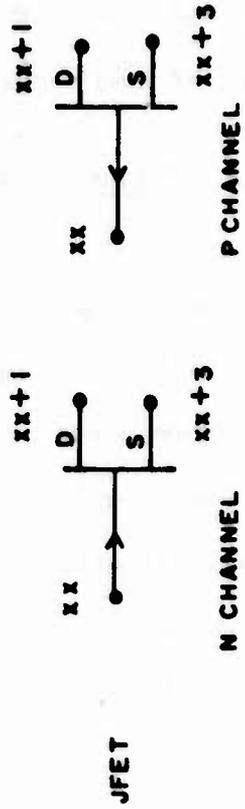
VACUUM TUBE



SOLID STATE



NPN PNP



DEVICE MODELS

introductory purposes only. Detailed descriptions of node numbering conventions and parameters necessary for the complete definition of each circuit element are given in Section 3.

### 1.2.2 Circuit Excitation and Order of Analysis

The nonlinear transfer functions computed by NCAP are voltage transfer ratios which relate an output response voltage to one or more input excitation voltages. Therefore, in order to determine a nonlinear transfer function, it is necessary to define the parameters of the input signals of the circuit and the frequencies at which the analysis is to be performed.

In NCAP these input signals are considered to be generated by independent sinusoidal voltage sources. Voltage sources (generators) can be connected between any two nodes in the circuit, and a single source can generate an arbitrary number of frequencies. Frequency control and the method of embedding voltage sources in NCAP circuits is described in depth in Section 3.3.1.

At this point it is important to understand that the order of analysis which the program will carry out is equal to the total number of defined frequencies in the circuit and that nonlinear transfer functions will be computed for all of the  $2^n - 1$  possible combinations of the  $n$  input frequencies. For example, if there are three input frequencies in the circuit,  $f_1$ ,  $f_2$ , and  $f_3$ , then a third order analysis will be performed and seven nonlinear transfer functions will result:

<u>ORDER</u>	<u>TRANSFER FUNCTION</u>	<u>FREQUENCY COMBINATION</u>
1	$H_1(f_1)$	$f_1$
1	$H_1(f_2)$	$f_2$
1	$H_1(f_3)$	$f_3$
2	$H_2(f_1, f_2)$	$f_1 + f_2$
2	$H_2(f_1, f_3)$	$f_1 + f_3$
2	$H_2(f_2, f_3)$	$f_2 + f_3$
3	$H_3(f_1, f_2, f_3)$	$f_1 + f_2 + f_3$

### 1.3 Preliminary Data Preparation

Since the NCAP analysis is performed on a nodal basis, the first step in the analysis of a circuit should be the drawing of its complete circuit model. This diagram should include all of the NCAP elements which can be identified.

Next the circuit nodes must be identified and numbered. The signal ground is automatically defined by the system as node 0, and the numbering of the remaining nodes must begin with 1 and continue sequentially until all nodes have been numbered. There is no requirement that adjacent nodes be assigned sequential numbers, but no numbers in the sequence may be omitted. Such an omission will cause erroneous results.

As mentioned previously the assignment of nodes for NCAP elements which have more than two nodes is done automatically by the system. The automatic node numbering conventions are discussed in detail in Section 3 and are depicted in Table 1.

It is important to understand that automatic node numbering

of multi-node devices will affect the identification of nodes on the circuit diagram. For example, the vacuum triode is a three node device. Its location in the circuit is defined by the number of the grid node which is assigned by the user. According to the convention built into the triode model, if the grid node is numbered  $xx$ , then the plate and cathode nodes automatically become  $xx + 1$  and  $xx + 2$  respectively. Therefore, as the analyst identifies nodes on his circuit diagram, if he assigns number 2, for instance, to the grid node of a triode, he must be aware that the plate automatically becomes node 3 and the cathode node 4.

Once the nodes have been assigned the circuit can be "translated" into the appropriate NCAP statements and prepared for input to the computer.

#### 1.4 Optional Features of NCAP

A number of optional features have been incorporated in the NCAP system to increase its versatility and ease-of-use. For example, in order to provide the user with a method of analyzing circuits over a range of frequency or linear component values, an incremental sweep capability has been included in the NCAP program. This feature enables the user to specify numerous analyses of a given circuit in a single computer run. The basic circuit description, together with all sweep definitions, is input only once. The system then automatically re-analyzes the circuit for all possible frequency and component values. In-depth discussions of frequency and linear element sweep

capabilities are presented in Sections 3.3 and 3.4.

A modify feature, which allows the user to alter nonlinear device parameters and re-analyze a circuit in a single computer run, has also been incorporated in NCAP. Such modification may also be applied to frequency and component values, either to change the parameters of a previously defined sweep, or to define additional values which may lie outside the range of a sweep. Modification is covered in Section 3.6.

Because a typical NCAP computer run can result in a large volume of printed output, several features have been included in the system to allow the user to control the volume of printed output as well as to specify plots of frequency vs. transfer function at selected nodes and orders. Output control is described in detail in Section 3.7.

SECTION 2

The first section of the report discusses the structure of the NCAP input language. It describes the various components and their interactions. The second section details the syntax and semantics of the language. The third section provides examples of input and output. The fourth section discusses the implementation and performance. The fifth section concludes the report and suggests future work.

**SECTION 2**

**STRUCTURE OF THE NCAP INPUT LANGUAGE**

The structure of the NCAP input language is defined by a set of rules and conventions. These rules govern the format and content of the input data. The language is designed to be flexible and extensible, allowing for the inclusion of new data types and structures. The input is processed by a parser that converts the raw data into a structured format. This structured format is then used by the system to perform various operations and generate output. The language is intended to be user-friendly and easy to learn, with clear documentation and examples provided. The structure of the language is a key component of the overall system architecture and is essential for the efficient processing of input data.

## SECTION 2: STRUCTURE OF THE NCAP INPUT LANGUAGE

The user communicates with the system by means of the NCAP input language. This language consists of a set of statements which allow the analyst to easily define the circuit configuration, its frequencies and order of analysis, and to control the output format. The input specifications (or language statements) are punched on cards or an equivalent time-sharing terminal. For purposes of this discussion the input medium is assumed to be punched cards.

### 2.1 Control Statements

A typical NCAP input deck consists of a series of control statements which define circuit topology, circuit excitation, linear and nonlinear elements, solution modification and output. Each control statement is contained on a single card and begins with an asterisk (\*) which identifies it to the system as a control statement. Following the asterisk are certain keywords which identify the function of the statement. These \* control statements are command-oriented: they are used to instruct the system to perform a particular function or to initiate the definition of a circuit element. For example, to begin a circuit description the analyst uses the statement:

\* START CIRCUIT

To begin the definition of a transistor he uses:

\* TRANSISTOR

To specify plotted output the statement is:

\* PLOT

## 2.2 Data Statements

Many of the \* control statements require additional input, such as node numbers or device parameters, to complete the definition of their function. This additional information is supplied by data statements which directly follow the \* control statement with which they are associated. Data statements do not begin with an asterisk since this character is reserved for the purpose of identifying control statements. Some data statements contain keywords, such as NODE, which are used for identification. Others contain only data values and are identified by their position within the input deck.

A complete description of each \* control statement and its associated data statements is given in Section 3.

## 2.3 Input Format

In general, NCAP input data is expressed in free-form. This eliminates the need for counting card columns or for placing individual data values in specific card fields. Free-form was selected because it is both easy to use and tends to minimize user errors.

There are two general rules which govern the use of free-form input:

1. The blank (or space) is used as a delimiter. That is,

successive data values are separated on a card by one or more blanks. This rule implies that no blanks may be embedded within a data value. For example, if the desired input value is 3.1416, those six characters must occupy contiguous card columns.

2. Since the blank is used as a delimiter it cannot be used interchangeably with the digit zero. All zero values must be explicitly punched on input cards as zero.

#### 2.4 Data Types

There are three types of data values used in the NCAP language.

1. Alphabetic - a string of alphabetic characters

EXAMPLES: TRANSISTOR  
          NODE

2. Integer - a whole number, expressed as a string of decimal digits. The decimal point is always omitted but is assumed to be to the right of the least significant digit. Negative values are preceded by a minus sign, while unsigned values are assumed to be positive. Integers have up to ten digits of accuracy.

EXAMPLES: 175  
          -3  
          16777216

3. Decimal - a floating point number expressed in one of two ways:

- a) A string of one to nine signed or unsigned decimal digits written with a decimal point.

EXAMPLES: 75.  
123.64  
-5.6

- b) One to nine decimal digits with a decimal point followed by a decimal exponent. The exponent is expressed as the letter E followed by a signed or unsigned integer. The exponent may be explicitly zero but cannot be blank. A decimal value has a precision of eight digits. Its magnitude must lie between  $10^{-38}$  and  $10^{38}$ .

EXAMPLES:

7.0E2 7.E02 .7E3

(All mean  $7 \times 10^2$  or 700.0)

-7.0E-3 -.7E-02 -7.0E-03

(All mean  $-7 \times 10^{-2}$  or -.007)

## 2.5 Input Error Detection

As the input deck is read and processed by the NCAP program, each of the user's statements is rigorously checked for possible errors. As a part of the program's standard output format, images of all input cards are reproduced on the computer's line printer. In the event that an error is detected, a descriptive error message is printed after the card image on which the error appears. These messages are self-explanatory and contain sufficient information for the user to make appropriate changes in his deck.

After the entire input deck has been read and processed, if any errors have been detected, execution of the program will be terminated and the output will consist of only input card images and pertinent error messages. If no errors are found, execution will continue into the computational phases of the program.

**SECTION 3**

**DETAILED DESCRIPTION OF NCAP STATEMENTS**

### SECTION 3: DETAILED DESCRIPTION OF NCAP STATEMENTS

As mentioned previously, a typical NCAP input deck consists of a series of \* control statements and their associated data statements.

The NCAP control statements tend to fall into seven categories according to their function within the total input stream. Therefore, for purposes of organization, this section is divided into seven subsections as follows:

- 3.1 Comment
- 3.2 Circuit Delineation
- 3.3 Sources
- 3.4 Linear and Nonlinear Components
- 3.5 Device Models
- 3.6 Solution Modification
- 3.7 Output Control

#### 3.1 Comment

The comment has been included among the NCAP \* control statements to aid the user in organizing or documenting his input deck. Comments provide a method for inserting brief remarks concerning the purpose or intent of statements which precede or follow them. The comment card diverges somewhat from the standard \* control card in that it begins with two \*'s. The \*\* may be followed by any desired text in the remaining card columns, but the entire comment must be contained on a single card. Long or multiple comments may be inserted in the deck by using multiple comment cards. An example of a comment card is:

```
** THIS IS A COMMENT
```

Comments are printed on the computer output but are not processed further by the program. A comment statement may be placed anywhere in the input deck and any number of comment cards may be used.

### 3.2 Circuit Delineation

The delineation statements initiate and terminate the description of the circuit to be analyzed. These statements and an explanation of their use follows.

#### \* START CIRCUIT

This statement is used as a delimiter which initiates the definition of the circuit. It indicates that all cards between it and the appearance of \* END CIRCUIT describe the circuit to be analyzed. No data statements are required.

#### \* END CIRCUIT

This statement is used as a delimiter to terminate the definition of the circuit. No data statements are required. Every circuit to be analyzed by NCAP must contain one and only one \* START CIRCUIT and \* END CIRCUIT pair.

#### \* END

This statement is the last card in any NCAP input deck. It indicates the end of all data and initiates the computational phase of the program. Every NCAP input deck must contain one and only one \* END statement.

### 3.3 Sources

#### 3.3.1 Independent Voltage Source

In order to determine a nonlinear transfer function, it is necessary to define the parameters of the input signals and the frequencies at which the analysis is to be performed. In NCAP these input signals are considered to be independent sinusoidal voltage sources. The complete definition of such a Thevenin voltage generator includes the nodes between which it is connected, its frequencies, peak amplitudes, and impedances. The definition of a generator is initiated by the following control statements:

\* GENERATOR

This statement indicates that all cards between it and the next \* control statement describe a Thevenin voltage generator. The required data statements are:

NODE      xx      yy

The NODE statement contains the word NODE followed by the integer node numbers xx and yy, which represent the nodes between which the generator is connected. There is no requirement that the two nodes be numbered sequentially. The generator may be grounded by defining yy to be zero.

The definition of a generator includes the specification of its frequencies. These are provided on FR data cards. There must be one FR data card for each frequency which a generator produces. Also there must be as many FR data cards as there are total defined frequencies. The user assigns each frequency a

number which the program uses in keeping track of frequency combinations during processing. If there are n frequencies, then there must be n-FR cards, each with a different frequency number. A present program restriction limits the total number of frequencies defined in a circuit to six. The FR card has the following general form:

FR        Number        Value

The characters FR identify the statement as a frequency definition. "Number" represents the user-assigned frequency number, an integer. "Value" is the decimal frequency value in Hertz.

EXAMPLE: FR        2        3.0E6

This statement defines a frequency number 2 of 3 MHz.

Associated with each frequency is the peak amplitude of its in-phase and quadrature voltages. These are provided on AMP data cards of the following general form:

AMP        Real        Imag

The characters AMP identify the statement as an amplitude definition. "Real" and "Imag" represent the (signed) decimal values of the peak amplitudes of the in-phase and quadrature voltages respectively.

EXAMPLE: FR        2        3.0E6  
          AMP        5.0        0.0

This sequence defines frequency number 2 generating 3 MHz with a 5 volt in-phase amplitude and no quadrature component.

The amplitude card must immediately follow the frequency card with which it is associated. As a default, if an AMP card

is omitted, a one-volt in-phase generator with no quadrature component will be assumed for the preceding frequency.

The final specification of a generator is its Thevenin impedances. These are provided on IMP cards which follow all the FR and AMP cards of a given generator. There must be one IMP card for each frequency combination at which the circuit is to be analyzed. For  $n$  defined generator frequencies, there will be  $2^n - 1$  frequency combinations, and therefore,  $2^n - 1$  impedances. The general form of the IMP card is as follows:

IMP	Real	Imag	Combination
-----	------	------	-------------

The characters IMP identify the statement as an impedance definition. "Real" and "Imag" represent the real and imaginary components of the generator impedance in decimal form. "Combination" represents an integer value which denotes the frequency combination at which the impedance is applicable. This value is determined as follows: If the frequency value under consideration is  $f_1 + f_2 + f_3$ , then the frequency combination is 123.

EXAMPLES:	IMP	80.	40.	1
	IMP	80.	40.	2
	IMP	70.	0.	3
	IMP	100.	100.	12
	IMP	75.	10.	13
	IMP	75.	10.	23
	IMP	70.	87.5	123

This sequence defines the impedances for all possible frequency combinations of a three-tone generator, where the frequencies are FR 1, FR 2, and FR 3.

As a default, if the generator impedance is constant for all frequency combinations, then only one IMP card need be used, and the combination parameter is left blank. This option is used

when a standard signal generator is the source. In this instance, the impedance is the generator's impedance, typically 50 ohms, and the single IMP card would be:

```
IMP      50.      0.
```

As a default for an "ideal" source, the value of the constant impedance should be as close to one as possible.

In summary, the complete definition of an independent voltage source requires: a \* GENERATOR card and a NODE data card; n-pairs of FR/AMP cards, one pair for each defined frequency; and  $2^n - 1$  IMP cards, one for each possible frequency combination. An example of a three-tone generator definition follows:

```
* GENERATOR
NODE      5          7
FR        1          500.E3
AMP       1.0        0.0
FR        2          500.E3
AMP       .707       .707
FR        3          -250.E3
AMP       .636       .772
IMP       80.        40.          1
IMP       80.        40.          2
IMP       70.        0.           3
IMP      100.       100.         12
IMP       75.        10.         13
IMP       75.        10.         23
IMP       70.        87.5        123
```

It is important to remember that the order of analysis which the program will carry out is equal to the total number of defined frequencies in the circuit (see Section 1.2.2). In the case where several generators are used, each with its own distinct frequencies, then the order of analysis will be the total number of all such frequencies for all generators.

In general,  $2^n - 1$  nonlinear transfer functions will be

computed for a circuit with n defined frequencies. Therefore, if n'th harmonic analysis is desired, there must be n defined frequencies in the circuit. Each FR data card will contain the fundamental frequency value, with each one being assigned a different frequency number. For example, a second harmonic analysis of a frequency of 1 MHz would require the following FR cards:

```
FR    1    1.0E6
FR    2    1.0E6
```

A third-order intermodulation of two tones at 1 MHz and 1.5 MHz, where the product of interest is  $2f_1 - f_2$ , would require the following:

```
FR    1    1.0E6
FR    2    1.0E6
FR    3   -1.5E6
```

By using an expanded version of the FR data statement the analyst may specify a frequency sweep. This feature allows the user to define multiple analyses of a given circuit over a range of generator frequency values in a single computer run. The general form of the swept FR statement is:

```
FR      Number      Start      Stop      Steps      Type
```

The characters FR identify the statement as a frequency definition. "Number" is the user-assigned frequency number described previously. "Start" and "Stop" represent the initial and terminal values of the frequency sweep in decimal form. "Steps" represents an integer value defining the number of times the frequency value is to be incremented (hence it defines the number of analyses to be performed). "Type" is an alphabetic

datum, either LIN or LOG, which indicates whether the increment is to be linear (additive) or logarithmic (multiplicative).

EXAMPLE: FR 3 1.0E6 5.0E6 5 LIN

This statement defines a sweep of frequency number 3 to begin at 1 MHz and proceed linearly through five analyses until the frequency value reaches 5 MHz.

It should be noted that the "Step" parameter defines the number of times the circuit is to be analyzed rather than the value of the increment. For linear sweeps the increment is automatically calculated by the program according to the expression:

$$\text{Increment} = \frac{\text{Stop} - \text{Start}}{\text{Steps} - 1}$$

For logarithmic sweeps the increment is:

$$\text{Increment} = \frac{\frac{\text{Stop}}{\text{Start}}}{\text{Steps}-1}$$

In determining the appropriate value for the "Step" parameter, the analyst should be aware that the Start and Stop values each count as a step. For example, if two analyses are desired, first at 1 MHz and then at 3.5 MHz, the FR sweep specification would be:

FR 3 1.0E6 3.5E6 2 LIN

It is possible to define several simultaneous frequency sweeps. For example, the sequence:

FR 1 1.0E6 5.0E6 5 LIN  
FR 2 -.5E6 -4.5E6 5 LIN

defines a simultaneous 5-step sweep in which the second-order frequency value ( $f_1 + f_2$ ) is the constant .5 MHz. Multiple frequency sweep specifications always result in simultaneous increments of the frequency values involved. That is, at the beginning of each analysis, all frequencies for which sweeps have been defined will be incremented simultaneously. Using the previous example as an illustration, the resulting frequency values used at each iteration (or analysis) are tabulated below:

<u>ITERATION</u>	<u>FR 1</u>	<u>FR 2</u>
1	1.0E6	-.5E6
2	2.0E6	-1.5E6
3	3.0E6	-2.5E6
4	4.0E6	-3.5E6
5	5.0E6	-4.5E6

When simultaneous frequency sweeps are used, each having a different "Step" parameter, then the largest defined "Step" parameter will determine the number of analyses to be performed. As the analyses proceed, each frequency value will be incremented until its "Stop" value has been reached. Once the Stop value has been reached, that frequency will remain constant until all defined sweeps have been satisfied. For example:

FR	1	1.0E6	6.0E6	6	LIN
FR	2	-.5E6	-3.5E6	4	LIN
FR	3	10.E6			

The first tone will assume six values from 1 MHz to 6 MHz; the second will assume four values from -.5 MHz to -3.5 MHz (notice that this sweep runs backwards); and the third tone, for which no sweep has been defined, will remain constant through all analyses at 10 MHz. The resulting frequency values for each iteration will be as follows:

<u>ITERATION</u>	<u>FR 1</u>	<u>FR 2</u>	<u>FR 3</u>
1	1.0E6	-.5E6	10.E6
2	2.0E6	-1.5E6	10.E6
3	3.0E6	-2.5E6	10.E6
4	4.0E6	-3.5E6	10.E6
5	5.0E6	-3.5E6	10.E6
6	6.0E6	-3.5E6	10.E6

The user should be cautious not to confuse the multiple analyses resulting from frequency sweeping with the order of analysis of a circuit. For each iteration of a frequency sweep, the nonlinear transfer functions are computed for all possible frequency combinations. In the case of the three-tone generator in the preceding example, six complete circuit analyses are performed, each at  $2^3 - 1 = 7$  possible frequency combinations. The output of such a computer run would contain 42 nonlinear transfer functions.

In general, by the appropriate use of frequency sweeping and constant frequency values, the analyst is able to create a wide variety of circuit excitations. Due to present program limitations, only one generator in a circuit may use swept frequencies. This restriction is discussed again in Section 3.6.2 which deals with the technique of frequency modification and suggests a possible method for circumventing this limitation.

One final point must be made concerning frequency sweeping.

It will be shown in Sections 3.4 and 3.6 that linear component sweeping and element modification result in numerous circuit analyses similar to those arising from frequency sweeps. In the case where frequency sweeping is used together with linear component sweeping and/or element modification, the frequency sweep is non-destructive. That is, the circuit is first analyzed over the entire range of defined frequency sweeps; if additional analyses are required due to linear component sweeps or element modification, all frequency sweeps are re-initialized at their starting values and executed again each time a circuit modification is encountered. In this way, circuits are analyzed for all desired element values over the complete frequency range.

### 3.3.2 Linear Dependent Sources

NCAP recognizes four types of linear dependent sources:

- 1) Voltage dependent Current source (VC)
- 2) Voltage dependent Voltage source (VV)
- 3) Current dependent Current source (CC)
- 4) Current dependent Voltage source (CV)

In NCAP language the definition of any linear dependent source begins with an \* control statement followed by an expanded NODE card:

```
* LINEAR DEPENDENT SOURCE
NODE      xx      yy      a      b
```

The \* LINEAR DEPENDENT SOURCE statement indicates that all cards between it and the next \* control card describe a linear dependent source. The NODE card differs from the standard node definition since two pairs of nodes must be specified in order to

define both the nodes of dependence as well as the nodes of connection of the source. The integer values xx and yy define the nodes upon which the source is dependent. The integer values a and b define the nodes between which the source is connected.

The definition of a linear dependent source includes the identification of the type of dependence and type of source as well as the real and imaginary parts of the complex scale factor by which the magnitude and phase angle of the source is to be altered. These data are input on a free form parameter card of the following general form:

$\left. \begin{array}{l} VC \\ VV \\ CC \\ CV \end{array} \right\}$	Real	Imag
---------------------------------------------------------------------	------	------

The alphabetic characters identify the type of dependence and source (V for voltage, C for current), where the first character represents the type of dependence and the second character represents the type of source. "Real" and "Imag" are the real and imaginary parts of the complex scale factor expressed in decimal format. At least one of these values must be nonzero.

Because of the inherent properties of complex numbers, care must be taken in selecting the proper values for the scale factor. For example, suppose a voltage dependent voltage source is to be connected between nodes 3 and 4, with dependence upon nodes 6 and 9. Furthermore, the voltage at (3,4) is to be 3 times the voltage at (6,9), while the phase angle at (3,4) is to be the same as that at (6,9). The following NCAP statements would be used:

**\* LINEAR DEPENDENT SOURCE**

NODE	6	9	3	4
VV	3.0		0.0	

If the scale factor were defined with Real = 3. and Imag = 3., the resulting voltage at (3,4) would be approximately 4.2 times that at (6,9), and the phase angle would be the phase angle at (6,9) plus the phase angle of the scale factor (in this case +45 degrees).

When current dependent sources (CC or CV) are being specified, a parallel resistor, capacitor, inductor combination through which the dependent current flows must be defined. These component values are input on R, C, and/or L parameter cards which follow the CC or CV card:

R	Value
C	Value
L	Value

The R, C, and L denote a resistor, capacitor, or inductor respectively. The "Value" parameter is a decimal number expressed in Ohms, Farads, or Henries. The current dependent linear source models require that at least one of these components must be nonzero. If the current does not flow through all three of the components then the unused components may be eliminated from the input deck.

A summary of the complete definition of the four types of linear dependent sources follows:

- 1) Voltage dependent Current source  
 \* LINEAR DEPENDENT SOURCE  
 NODE      xx      yy      a      b  
 VC          Real      Imag
  
- 2) Voltage dependent Voltage source  
 \* LINEAR DEPENDENT SOURCE  
 NODE      xx      yy      a      b  
 VV          Real      Imag
  
- 3) Current dependent Current source  
 \* LINEAR DEPENDENT SOURCE  
 NODE      xx      yy      a      b  
 CC          Real      Imag  
 R          Value  
 C          Value  
 L          Value
  
- 4) Current dependent Voltage source  
 \* LINEAR DEPENDENT SOURCE  
 NODE      xx      yy      a      b  
 CV          Real      Imag  
 R          Value  
 C          Value  
 L          Value

### 3.3.3 Nonlinear Dependent Sources

NCAP recognizes four types of nonlinear dependent sources:

- 1) Voltage dependent Current source (VC)
- 2) Voltage dependent Voltage source (VV)
- 3) Current dependent Current source (CC)
- 4) Current dependent Voltage source (CV)

In NCAP language the definition of a nonlinear dependent source begins with an \* control statement followed by an expanded NODE card:

```
* NONLINEAR DEPENDENT SOURCE
NODE      xx      yy      a      b
```

The \*NONLINEAR DEPENDENT SOURCE statement indicates that all cards between it and the next \* control card describe a nonlinear dependent source. The NODE card differs from the standard node

definition since two pairs of nodes must be specified in order to define both the nodes of dependence as well as the nodes of connection of the source. The integer values xx and yy define the nodes upon which the source is dependent while the integer values a and b define the nodes between which the source is connected.

The definition of a nonlinear dependent source must include the identification of the type of dependence and the type of source as well as the power series coefficients which describe the nonlinearity. These data are input on a free-form parameter card of the following general form:

VC	}	Coef <sub>1</sub>	Coef <sub>2</sub>	.....	Coef <sub>10</sub>
VV					
CC					
CV					

The alphabetic characters identify the type of dependence and source (V for voltage, C for current), where the first character represents the type of dependence and the second character represents the type of source. The remaining parameters are the power series coefficients of the nonlinearity expressed in decimal form. From one to ten coefficients may be input, and they can be continued on additional cards from which the alphabetic designator is eliminated. Zero coefficient values must be explicitly defined.

When nonlinear current dependent sources (CC or CV) are being specified, a parallel resistor, capacitor, inductor combination through which the current flows must be defined. These component values are input on R, C, and/or L parameter

cards which follow the CC or CV card:

R	Value
C	Value
L	Value

The R, C, and L denote a resistor, capacitor, or inductor respectively. The "Value" parameter is a decimal number expressed in Ohms, Farads, or Henries. The current dependent nonlinear source models require that at least one of these components must be nonzero. If the current does not flow through all three of the components, then the unused components may be eliminated from the input deck.

Some examples of nonlinear dependent sources follow.

```
* NONLINEAR DEPENDENT SOURCE
NODE      6      9      3      4
VV      .63E-3      .55E-3      .47E-3      .31E-3
      .25E-3
```

This sequence defines a nonlinear voltage dependent voltage source located at nodes (3,4) and dependent on nodes (6,9). Its power series is to be carried out to fifth order. Note that the fifth coefficient is placed on a continuation card from which the alphabetic designation VV has been omitted.

```
* NONLINEAR DEPENDENT SOURCE
NODE      4      6      5      3
CV      5.3E-3      0.0      3.6E-3      2.9E-3
R      1.7E3
L      .01
```

This sequence defines a current dependent voltage source located at nodes (5,3) and dependent on nodes (4,6). Its power series is to be carried out to fourth order, but the second order coefficient is zero. The dependent current flows through a parallel resistor/inductor combination. This source does not

involve a parallel capacitor.

### 3.4 Components

#### 3.4.1 Linear Components

NCAP recognizes three linear components: the resistor, capacitor, and inductor. Linear components are defined by the control statement:

**\* LINEAR COMPONENTS**

This statement is followed by a series of data statements which define the connection of specific resistors, capacitors, and/or inductors between pairs of circuit nodes. These data cards have the following general form:

R	}	xx	yy	Value
C				
L				

The R, C, or L denote a resistor, capacitor, or inductor. xx and yy represent the integer node numbers between which the linear component is connected. The "value" parameter is a decimal number expressed in Ohms, Farads, or Henries.

**EXAMPLE:**

```
* LINEAR COMPONENTS
C 1 2 6.6E-9
R 2 3 20.2E3
L 2 0 3.E-9
R 3 0 .01
```

The parameters on the R, C, and L data cards are expressed in free-form. The node yy may be the reference or ground node by explicitly numbering it zero. The node xx cannot be zero.

A single \*LINEAR COMPONENTS card sequence may be used to define as many as 50 linear components. Any number of such card sequences may be used in a circuit description. Their placement within the input deck is arbitrary.

By using an expanded version of the R, C, and L data cards, a linear component sweep(s) may be defined. This feature allows the analyst to specify multiple analyses of a circuit over a range of component values in a single computer run. The general form of the swept linear component data card is:

R	}	xx	yy	Start	Stop	Steps	Type
C							
L							

The R, C or L and the node parameters (xx and yy) are identical to those of the standard linear component definition. The "Start", "Stop", "Step", and "Type" parameters are the same as those used in frequency sweep specifications (see Section 3.3.1).

An example of a resistor sweep follows:

```
R  2    3    45.    55.    11    LIN
```

In this case, the circuit will be analyzed 11 times, with the resistor connected between nodes 2 and 3 varying linearly from 45 Ohms to 55 Ohms.

Several linear components may be swept simultaneously, with the largest "step" parameter determining the total number of circuit analyses to be performed. For example,

```
* LINEAR COMPONENTS
R  2    3    45.    50.    6    LIN
C  1    2    6.6E-9
L  2    0    3.E-9    5.E-9    3    LIN
```

A total of six circuit analyses would result with the following

element values:

<u>ITERATION</u>	<u>R(2,3)</u>	<u>C(1,2)</u>	<u>L(2,0)</u>
1	45.	6.6E-9	3.E-9
2	46.	6.6E-9	4.E-9
3	47.	6.6E-9	5.E-9
4	48.	6.6E-9	5.E-9
5	49.	6.6E-9	5.E-9
6	50.	6.6E-9	5.E-9

Since the resistor sweep has the largest "step" parameter, it determines the total number of circuit analyses. The capacitor, for which no sweep is defined, remains constant throughout all analyses at 6.6E-9. The inductor reaches its "stop" value at iteration 3 and remains constant at 5.E-9 for the remaining analyses.

Linear component sweeping is destructive. Once such a sweep has been satisfied, the value of the swept component will remain constant at its "stop" value for any subsequent circuit analyses. Section 3.6.1, which deals with linear component modification, presents a method for re-initiating component sweeps.

In some instances it may be desirable to vary both linear component and frequency values, resulting in several levels of sweeping. When this occurs, the total number of circuit analyses performed will be the product of the maximum frequency "steps" and maximum component "steps". Analyses will be performed across the range of the largest frequency sweep, using the "start" values for swept linear components. After the entire frequency sweep has been satisfied, component values will be incremented, frequency sweeps re-initialized, and the analyses performed again across the range of frequencies. This procedure of component increment/frequency initialization will continue until all

component sweeps have been completed.

For example, consider the following circuit definition:

```
* START CIRCUIT
* GENERATOR
NODE 1      0
FR   1      1.0E6   5.0E6   5   LIN
FR   2      3.0E6
IMP  50.     0.
* LINEAR COMPONENTS
C    1      2      6.6E-9
R    2      3      20.2E-3
R    3      0      10.      20.      2   LIN
.    .    .
.    .    . (Additional circuit elements)
.    .    .
* END CIRCUIT
* END
```

The resulting iterations and values would be:

<u>ITERATION</u>	<u>FR 1</u>	<u>R(3,0)</u>
1	1.E6	10.
2	2.E6	10.
3	3.E6	10.
4	4.E6	10.
5	5.E6	10.
6	1.E6	20.
7	2.E6	20.
8	3.E6	20.
9	4.E6	20.
10	5.E6	20.

Note that the total number of circuit analyses (iterations) performed is equal to the product of maximum frequency steps times maximum component steps ( $5 \times 2 = 10$ ). The values of frequency number 2 as well as  $C(1,2)$  and  $R(2,3)$ , for which no sweeps have been defined, remain constant throughout all analyses. In iterations 1 through 5 the frequency sweep is performed with the value of  $R(3,0)$  constant at its initial sweep value. At iteration 6, the frequency sweep has been satisfied. At this point  $R(3,0)$  is incremented, and the frequency sweep is initialized at its starting value of 1.E6. Iterations 6 through

10 reanalyze the circuit over the range of the frequency sweep for the new resistor value. At iteration 10, the resistor sweep has been satisfied and the analyses terminate.

### 3.4.2 Nonlinear Components

NCAP recognizes three nonlinear components: the resistor, capacitor, and inductor. Nonlinear components are defined by use of the control statement.

#### \* NONLINEAR COMPONENTS

followed by a series of data cards which define the connection of specific nonlinear resistors, capacitors, and/or inductors between pairs of circuit nodes. These data cards have the following general form.

$\left. \begin{array}{c} R \\ C \\ L \end{array} \right\}$	xx	yy	Coef <sub>1</sub>	Coef <sub>2</sub>	...	Coef <sub>10</sub>
------------------------------------------------------------	----	----	-------------------	-------------------	-----	--------------------

The R, C, or L denote a resistor, capacitor, or inductor respectively. xx and yy represent the integer node numbers between which the nonlinear component is connected. The remaining parameters are the decimal coefficients which describe the nonlinearity (See Volume I, Section II, Equations 2.3-1, 2.3-2, 2.3-3 for R, C, L respectively). From one to ten coefficients may be input and they may be continued on additional cards from which the R, C, or L designation is omitted. Zero coefficient values must be explicitly defined.

EXAMPLE.

\* NONLINEAR COMPONENTS

R	1	2	1.7E3	2.0E3	2.4E3	2.9E3
C	4	0	1.2E-6	1.4E-6	1.7E-6	2.1E-6
L	5	6	.21	.24	.29	.37

The parameters on the R, C, and L data cards are expressed in free-form. The node yy may be the reference or ground node by numbering it zero. The node xx cannot be zero.

A single \* NONLINEAR COMPONENTS card sequence may be used to define as many as 10 nonlinear components. Any number of such card sequences may be used in an NCAP circuit definition. Their placement within the input deck is arbitrary.

### 3.5 Device Models

Mathematical models for six devices have been incorporated in the NCAP program and are automatically called for by the appropriate \* control card.

The input card sequences which are required to define device models are similar to each other. They consist of an \* control statement, which identifies the desired element, a NODE card, which provides node numbering information, and one or more parameter cards, which provide the various values required to complete the definition of the device.

Specific data requirements for each device are presented in detail in the subsections which follow. The number of parameters needed varies with each device, but in general they are decimal values expressed in free-form. Successive parameters, separated by one or more blanks, may be input on as many cards as the user

feels are necessary. They must, however, be input in the order in which they are listed in the following subsections. It is important to remember that parameters whose values are zero must be explicitly input as zero.

Any number of device card sequences may be contained in a given circuit definition, and they may appear anywhere in the input deck.

### 3.5.1 Vacuum Diode

The vacuum diode model uses the generalized  $3/2$  power law. It is represented as a two-node device whose location in the circuit is defined by the user-assigned plate and cathode node numbers.

The card sequence necessary for the complete definition of a vacuum diode is:

```
* VACUUM DIODE
NODE      xx      yy
- 3 parameters -
```

The \* VACUUM DIODE statement indicates that all cards between it and the next \* control card describe a vacuum diode. The plate and cathode node numbers are specified on the NODE card by the integer values xx and yy. The two node numbers need not be sequential. The three parameters required to complete the definition of a vacuum diode are:

<u>PARAMETER</u>	<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>
1	G	Perveance
2	E <sub>B</sub>	Operating point plate voltage
3	C	Plate-cathode capacitance

### 3.5.2 Vacuum Triode

The vacuum triode model uses the generalized 3/2 power law. The node numbering convention for the triode is built into the model, such that its location in the circuit is defined by giving the number of the grid node, xx. The remaining nodes are automatically assigned by the program as follows:

```

xx + 1   Plate
xx + 2   Cathode

```

The card sequence necessary for the complete definition of a vacuum triode is:

```

* VACUUM TRIODE
NODE      xx
- 9 parameters -

```

The \* VACUUM TRIODE statement indicates that all cards between it and the next \* control statement describe a vacuum triode. The grid node is defined by the integer value xx on the NODE card. The nine parameters required to complete the definition of the vacuum triode are as follows:

<u>PARAMETER</u>	<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>
1	$G_o$	Perveance for grid voltage = 0
2	$\mu$	Control grid $\mu$ ; an amplification factor dependent upon geometry
3	$E_{C_{max}}$	Grid voltage for perveance = 0
4	$E_G$	Operating point grid voltage
5	$E_B$	Operating point plate voltage
6	$\phi$	Offset voltage due to space-charge effects
7	$C_{gk}$	Grid-cathode capacitance
8	$C_{pg}$	Plate-grid capacitance
9	$C_{pk}$	Plate-cathode capacitance

### 3.5.3 Vacuum Pentode

The vacuum pentode model uses the generalized  $3/2$  power law. The node numbering convention for the pentode is built into the model, such that its location in the circuit is defined by giving the number of the grid node,  $xx$ . The remaining nodes are automatically assigned by the program as follows:

$xx + 1$	Screen
$xx + 2$	Plate
$xx + 3$	Cathode

The card sequence necessary for the complete definition of a vacuum pentode is:

\* VACUUM PENTODE

NODE       $XX$

- 12 parameters -

The \* VACUUM PENTODE statement indicates that all cards between it and the next \* control statement describe a vacuum pentode. The grid node is defined by the integer value  $xx$  on the NODE card. The 12 parameters required to complete the definition of the vacuum pentode are as follows:

<u>PARAMETER</u>	<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>
1	$G_o$	Perveance for grid voltage = 0
2	$\mu$	Control grid $\mu$ ; an amplification factor dependent upon geometry
3	D	Division constant; see note below

4	m	Constant $\ll 1$ ; on the order of 0.2
5	$E_{G1}$	Operating point grid voltage
6	$E_{C_{max}}$	Grid voltage for perveance = 0
7	$\phi$	Offset voltage due to space-charge effects
8	$E_{G2}$	Operating point screen voltage
9	$E_B$	Operating point plate voltage
10	$C_{gk}$	Grid-cathode capacitance
11	$C_{pg}$	Plate-grid capacitance
12	$C_{pk}$	Plate-cathode capacitance

Note: The division constant, D, is defined as follows:

$$D = \frac{i_b}{i_2} \quad \left| \quad e_b = e_2 \right.$$

where,  $i_b$  = plate current  
 $i_2$  = screen current  
 $e_b$  = plate voltage  
 $e_2$  = screen voltage

### 3.5.4 Semiconductor Diode

The semiconductor diode is an internodal device, whose location in the circuit is defined by the user-assigned node numbers  $xx$  and  $yy$ . The semiconductor diode may be either forward-biased or reverse-biased. Therefore it has two representations in the NCAP language. For either type, however, a standard \* control and NODE card sequence is used:

```
*SEMICONDUCTOR DIODE
NODE      xx      yy
```

The \* SEMICONDUCTOR DIODE statement indicates that all cards between it and the next \* control card describe a semiconductor diode. The plate and cathode node numbers are defined by the integer values  $xx$  and  $yy$  on the NODE card. The two nodes need not be numbered sequentially. The device may be grounded by explicitly defining node "yy" to be zero. Node "xx" cannot be zero.

The forward-biased diode is represented by an exponential resistive nonlinearity in parallel with a diffusion capacitance. To complete the definition of a forward-biased semiconductor diode, an FB data card is required, followed by four parameters. The characters FB on the data card indicate that the forward biased option is to be used. The four parameters are as follows:

<u>PARAMETER</u>	<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>
1	$I_D$	Bias current
2	$n$	Non-ideality factor
3	$C_j$	Forward-biased junction

capacitance, extrapolated  
to zero current

4

$C'_j$

Derivative of junction bias  
with respect to current

The reverse-biased diode is represented by a varactor, or nonlinear capacitor, in parallel with a resistor. To complete the definition of a reverse-biased semiconductor diode, an RB data card is required, followed by four parameters. The characters RB indicate that the reverse-biased option is to be used. The four parameters are as follows:

<u>PARAMETER</u>	<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>
1	$V_D$	Magnitude of the bias voltage
2	K	Varactor capacitance at 1 volt reverse bias
3	$\mu$	See note
4	R	Reverse-bias leakage resistance

Note: The parameters V and MU are related to the varactor capacitance,  $C(V)$ , by the following expression:

$$C(V) = KV_D^{-\mu}$$

### 3.5.5 Bipolar Junction Transistor

The bipolar junction transistor is modeled as a nonlinear T. The node numbering convention for the transistor is built into the model, such that its location in the circuit is defined by giving the number of the external base terminal, xx. The remaining nodes are automatically assigned by the program as follows.

xx+1      Internal Junction  
 xx+2      External Collector  
 xx+3      External Emitter

The card sequence necessary for the complete definition of a bipolar junction transistor is:

\* TRANSISTOR

NODE      XX

- 16 parameters -

The \* TRANSISTOR statement indicates that all cards between it and the next \* control statement describe a bipolar junction transistor. The external base node is defined by the integer value xx on the NODE card. The sixteen parameters required to complete the definition of the transistor are as follows:

<u>PARAMETER</u>	<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>
1	n	Avalanche exponent
2	$V_{CB}$	Collector-base bias voltage
3	$V_{CBO}$	Avalanche voltage
4	$\mu$	Collector capacitance exponent
5	$I_C$	Collector bias current
6	$I_{C_{max}}$	Collector current at maximum D.C. current gain
7	a	$h_{FE}$ nonlinearity coefficient
8	$h_{FE_{max}}$	Maximum D.C. current gain
9	K	Collector capacitor scale factor
10	n	Diode nonideality factor

11	$C_{je}$	Base-emitter junction space charge capacitance
12	$C'_2$	Derivative of base-emitter diffusion capacitance
13	$r_b$	Base resistance
14	$r_c$	Collector resistance
15	$C_1$	Base-emitter capacitance
16	$C_3$	Base-collector and overlap capacitance

### 3.5.6 Junction Field Effect Transistor

The junction field effect transistor (JFET) is a three-node device. The node numbering convention for the JFET is built into its model such that the location of the device in a circuit is defined by giving the number of the gate node,  $xx$ . The remaining nodes are automatically numbered by the program as follows:

```

xx + 1    Drain
xx + 2    Internal Node
xx + 3    Source

```

The JFET nonlinearities may be expressed either as power series coefficients or as parameters for the analytic generation of the device's nonlinearities. Consequently, the JFET has two representations in the NCAP language. For either option, however, the following card sequence must be used:

```

*JFET
NODE      xx
CONSTANTS      CGD      RS

```

The \*JFET statement indicates that all cards between it and the next \*control statement describe a junction field effect transistor. The gate node is defined by the integer value xx on the NODE card. The CONSTANTS (this may be abbreviated to CO) data card alerts the program that the two decimal values to follow are the constants  $C_{GD}$  (gate-drain capacitance) and  $R_S$  (source resistance).

If the power series coefficients are to be input (rather than analytically generated) they are defined on BP and CP data cards which have the following general form:

```

PP  B1    B2    ...B10
CP  C1    C2    ...C10

```

The characters BP or CP indicate that the decimal values which follow are B or C parameters respectively. The BP or CP is followed by from one to ten B or C decimal coefficients expressed in free-form. The coefficient values may all be placed on the BP or CP card or they may be continued on additional cards, in which case the characters BP or CP are omitted from the continuation cards.

The B parameters are experimentally determined power series coefficients in the equation:

$$-i_d R_L = V_d = \sum_{m=1}^{\infty} B_m R_L V_g^m$$

where  $V_d$  = drain voltage  
 $i_d$  = drain current  
 $R_L$  = small resistive load

The C parameters are power series coefficients which characterize the nonlinear gate to source capacitance,  $C_{GS}$ . These C coefficients can be determined by measuring the gate-source capacitance as a function of the gate-source voltage, and performing a power series expansion around the operating point.

In summary the complete definition of a junction field effect transistor, for which the nonlinear coefficients are input, requires the following NCAP card sequence:

```

* JFET
NODE      xx
CONSTANTS  CGD  RS
BP        - one to ten B coefficients -
CP        - one to ten C coefficients -
    
```

If the analytic generation of coefficients is desired, an AC data card followed by eight decimal parameters is used. The characters AC indicate that the decimal values which follow are the parameters required for the analytic generation of coefficients. The eight parameter values may all be placed on the AC card or they may be continued on additional cards, in which case the characters AC are omitted from the continuation cards. The required parameters are defined as follows:

<u>PARAMETER</u>	<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>
1	$I_{D_{max}}$	Drain current at maximum D.C.
2	$\rho$	Saturated drain current parameter

3	$V_p$	Pinch-off voltage
4	$\psi$	JFET barrier potential
5	$V_{GS}$	External gate-source voltage
6	K	See note
7	m	Asymptotic slope of $C(V'_{GS})$ on logarithmic plot. See note
8	$V_o$	Gate-source capacitor's built in voltage

Note: The gate-source capacitance,  $C_{GS}$ , can be represented analytically by the equation:

$$C(V'_{GS}) = K (|V_o + V'_{GS}|)^{-m}$$

where the zero-bias value of the capacitance is  $K/|V_o|^{-m}$ .

In summary, the complete definition of a junction field effect transistor (for which the nonlinear coefficients are to be analytically generated) requires the following NCAP card sequence:

```

* JFET
NODE      xx
CONSTANTS  CGD  RS
AC        -eight parameters -

```

### 3.6 Solution Modification

In many modeling applications it is desirable to rerun a circuit analysis with certain element values changed to determine the circuit sensitivity to element variation. In the NCAP system, solution modification is performed by means of the modify feature. This allows certain element values to be changed and the circuit automatically re-analyzed. Modification is initiated by the following control statement:

\* MODIFY

The \* MODIFY statement indicates that all cards between it and the next \* control card define a modification of the element defined in the previous \* control sequence. Therefore, the \*MODIFY card must immediately follow the definition of the circuit element it affects. It in turn is followed by R, C, or L data cards for linear or nonlinear component modification, FR and AMP data cards for frequency modification, and device parameter cards for nonlinear element modification.

In order to redefine a given circuit element several times, its original definition may be followed by several \* MODIFY card sequences. Multiple modifications are performed sequentially. For example, if a transistor definition is followed by two modifies, three complete circuit analyses will be performed: first with the original transistor, then with the transistor defined by the first \* MODIFY, and finally with the transistor defined by the second \* MODIFY.

If more than one circuit element is followed by modify

specifications, all such elements will undergo modification simultaneously according to the first set of modifies, then according to the second set and so forth.

Solution modification is used only for changing circuit element values. Circuit topology, or the interconnection of elements between nodes, cannot be altered. Furthermore, in order for an element to be modified, it must have been defined in the original circuit. New elements cannot be added by using the modify feature.

### 3.6.1 Linear Component Modification

An \* MODIFY card followed by R, C, and/or L data cards will cause linear component values defined in the previous \* LINEAR COMPONENTS sequence to be modified. The contents of the R, C, or L data cards in the modify sequence must be identical to those in the original circuit definition, except that component values are changed to reflect the desired modification(s):

```
* LINEAR COMPONENTS
R   1   2   2.0E3
C   2   3   6.6E-9
L   2   0   3.E-9
* MODIFY
R   1   2   5.0E3
```

The original circuit definition includes a 2K resistor between nodes 1 and 2. The \* MODIFY sequence indicates that the original circuit has been completely analyzed, the resistor value is to be changed to 5K and the circuit re-analyzed.

Any or all of the components defined in the \* LINEAR

COMPONENTS sequence can be redefined with a single modification. Components which are to remain constant need not be included in the \* MODIFY sequence. For example:

```
* MODIFY
C      2      3      10.E-9
L      2      0      7.0E-9
```

would change both the capacitor and inductor, while:

```
* MODIFY
R      1      2      5.0E3
C      2      3      10.E-9
L      2      0      7.0E-9
```

would cause redefinition of all linear components defined in the previous \* LINEAR COMPONENTS sequence.

Multiple modifies may follow a single linear components definition:

```
* LINEAR COMPONENTS
R      1      2      2.0E3
C      2      3      6.6E-9
L      2      0      3.0E-9
* MODIFY
R      1      2      5.0E3
* MODIFY
C      2      3      10.E-9
L      2      0      7.0E-9
```

This sequence would produce three circuit analyses, with the two modifications being performed sequentially. Component values for each iteration are tabulated below:

<u>ITERATION</u>	<u>P(1,2)</u>	<u>C(2,3)</u>	<u>L(2,0)</u>
1	2.0E3	6.6E-9	3.0E-9
2	5.0E3	6.6E-9	3.0E-9
3	5.0E3	10.E-9	7.0E-9

If linear component sweeping and modification are used together, a variety of results may be obtained. The \* MODIFY sequence can be used to hold a previously swept component constant or vice versa:

```

* LINEAR COMPONENTS
R   1   2   1.0E3   10.E3   10   LIN
C   2   3   6.6E-9
* MODIFY
R   1   2   50.
C   2   3   6.0E-9   7.E-9   11   LIN

```

The original component definition includes a resistor sweep and a constant capacitor value. The modification causes the previously swept resistor to remain constant while the capacitor sweeps.

Modification can also be used to redefine linear component sweeps, to change the value of the increment, or to skip over particular element values which are not of interest in the analysis:

```

* LINEAR COMPONENTS
R   1   2   1.E3   10.E3   10   LIN
* MODIFY
R   1   2   20.E3  100.E3   5   LIN

```

In this case, the resistor first sweeps from 1K to 10K in ten steps, then from 20K to 100K in 5 steps.

Since linear component sweeping is destructive, any circuit analyses which occur after such a sweep has been satisfied, will treat the value of the subject component as a constant. For example:

```

*LINEAR COMPONENTS
R   1   2   1.E3   10.E3   10   LIN
C   2   3   6.6E-9
* MODIFY
C   2   3   10.E-9

```

In the original definition, the resistor sweeps from 1K to 10K, with the capacitor value remaining constant at 6.6E-9. After the tenth analysis has been completed and the resistor sweep satisfied, the modification is performed.

The result is a single re-analysis, with the resistor at 10K

(the "stop" value of the sweep) and the capacitor at the modified value of 10.E-9.

With the proper use of the modification feature, the analyst can reinstate a previous linear component sweep so as to override its destruction. For example:

```

* LINEAR COMPONENTS
R   1   2   1.E3   10.E3   10   LIN
C   2   3   6.6E9
* MODIFY
R   1   2   1.E3   10.E3   10   LIN
C   2   3   10.E-9

```

This card sequence would result in 20 analyses in which each capacitance is analyzed for each of the ten resistive values.

### 3.6.2 Nonlinear Component Modification

An \* MODIFY card followed by R, C, and/or L data cards will cause nonlinear component power series coefficients defined in the previous \* NONLINEAR COMPONENTS sequence to be modified. The contents of the R, C, and L data cards in the modify sequence must be identical to those in the original circuit, except that the coefficient values are changed to reflect the desired modification(s):

```

* NONLINEAR COMPONENTS
R   1   2   1.7E3   2.0E3   2.4E3   2.9E3
C   4   0   1.2E-6   1.4E-6   1.7E-6   2.1E-6
L   6   4   .21     .24     .29     .37
* MODIFY
L   6   4   .35     .37     .41     .45

```

In the original circuit the inductor between nodes 6 and 4 is defined in terms of the fourth order power series coefficients .21, .24, .29, and .37. The \* MODIFY sequence indicates that after the original circuit has been analyzed, the coefficient

values for the inductor are to be changed to .35, .37, .41, and .45 and the circuit re-analyzed.

As with linear components, any of all of the components defined in the \* NONLINEAR COMPONENTS sequence can be redefined with a single modification. Components which are to remain constant need not be included in the \* MODIFY sequence (see Section 3.6.1).

### 3.6.3 Frequency Modification

An \* MODIFY card followed by FR and AMP data cards will cause frequency and amplitude values defined in the previous \*GENERATOR sequence to be modified. Impedance values cannot be modified.

An example of a frequency modification follows:

```

* GENERATOR
NODE      1      0
FR        1     10.E6
FR        2     -5.E6
IMP       50.    0.
* MODIFY
FR        2     20.E6
AMP       5.     0.

```

The original generator definition specifies two tones, at 10 MHz and -5 MHz, each using the default amplitude. In the modified generator, frequency number 1 remains unchanged, while frequency number 2 changes to 20 MHz with an amplitude of 5 volts in-phase and zero quadrature component.

Any or all of the frequencies and amplitudes (whether explicit or default) defined in a given \* GENERATOR sequence can be changed with a single modify specification. The modified FR cards must be identical to those appearing in the original generator definition, except that the frequency values are changed to reflect the desired modifications. Frequencies and amplitudes which are to remain unchanged need not appear in the \*MODIFY sequence.

Since FR and AMP data cards are used in pairs, if an amplitude is to be modified, both the FR and AMP cards must appear in the modify sequence. If only the frequency value is to be changed, the associated AMP card may be omitted. For example:

```

* GENERATOR
NODE      1      0
FR        1     10.E6
FR        2      5.E6
AMP       5.     0.
IMP       50.    0.
* MODIFY
FR        1     10.E6
AMP       5.     0.
FR        2     20.E6

```

In this case, although the frequency value for FR 1 does not change, in order to modify its amplitude, both the FR 1 and AMP cards must appear in the modify sequence. For frequency number 2, since only the frequency value is to be modified the AMP card is omitted from the modify sequence.

Frequency sweeping and modification may be used together. The modify can be used to sweep a frequency value which was previously held constant or vice versa:

```

* GENERATOR
NODE      1      2
FR        1      10.E6      50.E6      5      LIN
FR        2      5.E6
IMP       50.      0.
* MODIFY
FR        1      100.E6
FR        2      30.E6      60.E6      3      LIN

```

In the original generator definition, frequency number 1 sweeps while frequency 2 remains constant. The modified generator redefines the frequencies such that FR 1 remains constant while FR 2 sweeps.

With the appropriate use of frequency modification, the analyst may redefine frequency sweeps in order to vary the increment or to skip over particular frequency values:

```

* GENERATOR
NODE      1      0
FR        1      10.E6      30.E6      3      LIN
IMP       50.      0.
* MODIFY
FR        1      35.E6      50.E6      4      LIN
* MODIFY
FR        1      70.E6      100.E6      4      LIN

```

The original generator definition will result in analyses at 10, 20 and 30 MHz. The first modification changes the increment of the sweep and results in analyses at 35, 40, 45, and 50 MHz. The

second modification effectively skips over all values between 50 and 70 MHz and initiates a final sweep for 70, 80, 90 and 100 MHz.

Frequency modification is always applied after all frequency and linear component sweeps have been satisfied and after all other modifies have been completed.

```

* GENERATOR
NODE      1      0
FR        1    10.E6    20.E6    2      LIN
IMP       50.     0.
* MODIFY
FR        1    50.E6
* LINEAR COMPONENTS
R         2      3      1.E3    3.E3    3      LIN
C         3      0      6.6E-9
* MODIFY
C         3      0    10.E-9

```

In this case the following sequence of analyses would result:

ANALYSIS	FR 1	R(2,3)	C(3,0)
1	10.E6	1.E3	6.6E-9
2	20.E6	1.E3	6.6E-9
3	10.E6	2.E3	6.6E-9
4	20.E6	2.E3	6.6E-9
5	10.E6	3.E3	6.6E-9
6	20.E6	3.E3	6.6E-9
7	10.E6	3.E3	10.E-9
8	20.E6	3.E3	10.E-9
9	50.E6	3.E3	10.E-9

Note that while the resistor sweep is in effect (iterations 1 - 6), each new resistor value causes an initialization and restart of the frequency sweep, so that each resistor value is analyzed at every possible frequency value. At iteration 7 the resistor sweep is complete, the capacitor modification is applied, and the frequency sweep restarted. Since the resistor sweep is not redefined by a subsequent modify, the resistor value remains constant at 3K following the completion of the sweep. At

iteration 9, all defined frequency and linear component sweeps and element modifications have been completed and the frequency modification takes place.

For circuits which contain more than one generator, each of which undergoes modification, the frequency modifications will be performed simultaneously in sets:

```

* GENERATOR
NODE    1      0
FR      1    10.E6
IMP     50.    0.
* MODIFY
FR      1    30.E6
* MODIFY
FR      1    50.E6
* GENERATOR
NODE   10     12
FR      2    40.E6
IMP     75.    0.
* MODIFY
FR      2    60.E6

```

After the original circuit has been analyzed for FR 1 at 10 MHz and FR 2 at 40 MHz, the first set of modifications will be performed resulting in an analysis with FR 1 at 30 MHz and FR 2 at 60 MHz. At this point the modifications for FR 2 have been exhausted and its value remains constant at 60 MHz. FR 1 is modified again and a third circuit analysis is performed with FR 1 at 50 MHz and FR 2 at 60 MHz.

Since the analyst is restricted to a single generator with swept frequencies, care must be taken in the use of frequency modification for circuits which contain more than one generator:

```

* GENERATOR
NODE    1      0
FR      1    10.E6    40.E6    4    LIN
IMP     50.    0.
* GENERATOR
NODE   10     12
FR      2    50.E6

```

```

IMP      75.0      0.
* MODIFY
FR       2      60.E6   100.E6   5      LIN

```

This modification is illegal because it would result in two generators having swept frequencies simultaneously. In an actual NCAP computer run, if such a circuit were input, an error message indicating the double sweep would be printed and execution of the program would terminate.

On the other hand, by the proper application of frequency modification, one generator sweep can be "turned off" allowing the other generator to assume a sweep legally:

```

* GENERATOR
NODE     1      0
FR       1     10.E6   40.E6   4      LIN
IMP      50.     0.
* MODIFY
FR       1     40.E6
* GENERATOR
NODE    10     12
FR       2     50.E6
IMP      75.     0.
* MODIFY
FR       2     60.E6   100.E6   5      LIN

```

While the original generator definitions are in effect, the first generator sweeps while the second remains constant. The first modify turns off the sweep for frequency number 1 holding it constant at 40 MHz. It is then possible to modify the second generator to sweep without violating the multiple generator sweep rule. The resulting values for each iteration of frequency are tabulated below:

<u>ITERATION</u>	<u>FR 1</u>	<u>FR 2</u>	<u>FR 1 + FR 2</u>
1	10.E6	50.E6	60.E6
2	20.E6	50.E6	70.E6
3	30.E6	50.E6	80.E6
4	40.E6	50.E6	90.E6
5	40.E6	60.E6	100.E6
6	40.E6	70.E6	110.E6

7	40.E6	80.E6	120.E6
8	40.E6	90.E6	130.E6
9	40.E6	100.E6	140.E6

Iterations 1 through 4 are the result of the sweep of FR 1. After this sweep is satisfied the two frequency modifications are applied simultaneously at iteration 5. The modifications force FR 1 to be held constant at 40 MHz, enabling FR 2 to assume a frequency sweep. Iterations 5 through 9 are the result of the FR 2 sweep.

Note that since there are two defined frequencies in the circuit, the order of analysis is 2, and there are  $2^2 - 1 = 3$  possible frequency combinations for each sweep iteration. Therefore, this example would result in a total of 27 nonlinear transfer functions.

### 3.6.4 Dependent Source Modification

An \* MODIFY card followed by dependent source parameters will cause the parameter values of the dependent source defined in the previous \* LINEAR DEPENDENT SOURCE or \* NONLINEAR DEPENDENT SOURCE card sequence to be modified.

```
* LINEAR DEPENDENT SOURCE
NODE      6      9      3      4
CV        3.0    0.0
R         1.7E3
L         .01
* MODIFY
CV        4.0    0.0
R         3.5E3
C         2.5E-12
```

The \* MODIFY card will cause the scale factor of the current dependent voltage source located between nodes 3 and 4 to be

changed from (3.0, 0.0) to (4.0, 0.0). In addition, the value of the parallel resistor is to be changed from 1.7E3 to 3.5E3 and the parallel inductor is to be replaced by a 2.5E-12 farad capacitor.

It is important to understand that the node definitions or type of dependence or source cannot be altered by the modify feature. For this reason NODE data cards are never used in the \* MODIFY card sequence and the source type specification (VC, VV, CC, CV) on modify parameter cards must be the same as that in the original source definition.

### 3.6.5 Device Model Modification

An \* MODIFY card followed by device model parameter cards will cause the device in the previous \* control card sequence to be modified. All of the device models included in the NCAP system (see Section 3.5) can be modified. For example:

```

* TRANSISTOR
NODE      10
   5.0      8.76      200.      .209
 195.E-5    .006      1.14      75.
   0.0      1.0       0.0      2.80E-9
 400.      1.14E6    0.0      1.3E-12
* MODIFY
   4.6      8.86      140.      .348
  .0043    .150      .125    8.2
   0.0      1.0       0.0      2.80E-9
 400.      1.0E5    0.0      1.5E-12.

```

The \* MODIFY sequence will cause the parameters for the transistor at node 10 to be changed from their original values to those defined in the modify sequence.

Any or all of the device model parameters can be changed

with a single modify, but the entire data set must be input in the modify sequence regardless of the number which are actually changed by the modify.

It is important to understand that modification only affects device model parameter values and cannot be used to change node definitions or to otherwise alter the topology of the original circuit. For this reason, NODE data cards are never used in a modify card sequence.

Several modifications may follow the original definition of a device model in order to redefine that device several times. Such multiple modifications are performed sequentially.

When more than one device is to be redefined by modification, such modifications are performed simultaneously in sets. The devices are modified simultaneously according to the first set of modifies, then according to the second set and so forth.

### 3.7 Output Control

The output of a typical NCAP run, printed on the computer's line printer, contains a large volume of information. In general the output consists of images of all input cards, all circuit devices with their associated parameters values, and all scaled nonlinear transfer functions and node voltages. The transfer functions and node voltages are printed for each node and each order for every possible frequency combination, in both Cartesian and log polar form.

When the modify feature is used, the original circuit output is followed by a listing of all modified devices with their associated parameters and all scaled nonlinear transfer functions and node voltages for the modified circuit, one node to a line, for all frequency combinations, orders, and nodes.

In the event that errors are detected in the input deck, the printout of the erroneous input card will be followed by an error message describing the type of error encountered. Once such an error has been found, processing of the input deck will continue until the \* END card is read. At this point, execution of the program will terminate and the output will consist of only the input card images and appropriate error messages.

The successful analysis of a large circuit can result in an inordinately large amount of printed output. Therefore, several output control statements have been included in the NCAP language to allow the user to specify the desired output and to reduce the amount of printout obtained.

### 3.7.1 Suppressing Output of Circuit Element Data

The printing of any or all linear or nonlinear element data may be suppressed by the use of the following \* control statement:

\* PRINT SELECT

followed by a card containing either the word OFF or ON. The OFF option will cause print suppression of all circuit elements and their associated parameters which are defined in successive \*

control sequences, until an \* PRINT SELECT, ON option is encountered. The ON option causes printing of circuit elements and parameters defined in successive \* control statements to take place. Any number of \* PRINT SELECT, ON/OFF statements may be used, but their placement within the input deck is of great importance. If a single \* PRINT SELECT, OFF is placed at the beginning of the input deck, all linear and nonlinear element data will be print suppressed. By default, if no \* PRINT SELECT, OFF cards are used, all element data will be output.

On the other hand, if several ON/OFF options are appropriately placed within the input deck, the user can control exactly which elements are to be printed and which suppressed:

```
* PRINT SELECT
OFF
* LINEAR COMPONENTS
  .      .      (Linear component data)
  .      .
* PRINT SELECT
ON
* TRANSISTOR
  .
  .
  .
```

In this case, the printing of the components defined in the \*LINEAR COMPONENTS card sequence will be suppressed but the transistor data will be printed.

Generator data is always output. It cannot be suppressed by use of the \* PRINT SELECT feature.

### 3.7.2 Suppressing Output of Nodes and Orders

A slightly different form of the \* PRINT SELECT feature can be used to specify which nodes and/or orders are to be output while the others are print suppressed.

To print selected nodes the \* PRINT SELECT statement is followed by a NODE card with the following general form:

```
NODE    XX    YY    ZZ    .....
```

The word NODE identifies the statement as a node selection data card. The values XX,YY,ZZ... represent the integer node numbers for which nonlinear transfer functions and voltages are to be printed. Only one NODE card can be used, but it can contain up to ten node numbers. For example:

```
* PRINT SELECT  
NODE      1      3      5      11      14      22
```

would result in the printing of all nonlinear transfer functions and voltages at nodes 1, 3, 5, 11, 14, and 22; the output data for all other nodes in the circuit will not be printed.

To print selected orders, the \* PRINT SELECT statement is followed by an ORDER data card:

```
ORDER    XX    YY    ZZ    .....
```

The word ORDER identifies the statement as an order selection. The values XX, YY, ZZ ... represent the integer order numbers for which nonlinear transfer functions and node voltages are to be printed. Only one ORDER data card can be used, but it can contain up to ten order numbers:

```
* PRINT SELECT  
ORDER    1      3
```

would result in the printing of output data for first and third orders only.

If both nodes and orders are to be selected for output, the \* PRINT SELECT should be followed by both a NODE card and an ORDER card. This sequence may be placed anywhere in the input deck.

### 3.7.3 Plotted Output (Presently Being Developed)

Line printer plots of frequency vs. log polar transfer function may be obtained by use of the NCAP \* control statement:

\* PLOT

The \* PLOT indicates that all cards between it and the next \* control card define a plotted output.

The definition of a plot includes the frequency number whose values will appear on the abscissa of the plot and the node and order of the transfer function which will be used as the ordinate.

The desired frequency number is defined by an FR data card of the following general form:

FR      Number      Type

The characters FR identify the card as a frequency selection. "Number" is an integer value corresponding to the user-assigned number of a swept frequency defined in some generator in the circuit. "Type" is an alphabetic datum either LIN or LOG, which indicates whether the plot is linear or log scaled. "Type" must be the same as the type parameter used in the definition of the

frequency sweep under consideration: linear sweeps result in linear plots while logarithmic sweeps result in log plots. The frequency selected by the FR card must be a swept frequency since its range of values will constitute the range of values along the x-axis of the plot. A constant frequency would result in a single point rather than a curve.

The node selection is provided for by a NODE card:

NODE    XX

where XX is an integer node number defined in the circuit.

The order selection is provided for by an ORDER card:

ORDER    YY

where YY is an integer value which defines the order for which nonlinear transfer functions are to be plotted. For a circuit with n defined frequencies, the order selected for plotting must be either 1, (denoting first order) or n (denoting the maximum order of analysis of the circuit).

The user may specify headings to be printed above and/or below the plot for purposes of identification or documentation. The contents of these labels are provided on LABEL data cards.

LABEL    -up to 50 characters of text-

where the word LABEL is followed by up to 50 characters of text. If one LABEL card is used the text will appear above the plot. If two LABEL cards are used, the text from the first will be printed above the plot and the second below the plot. If only the lower label is desired, the text of the first LABEL card should be left blank. If both LABEL cards are omitted no headings will be printed.

In summary, the complete definition of a plot requires the following NCAP card sequence:

```
* PLOT
FR      Number      Type
NODE    XX
ORDER   YY
LABEL   -50 characters printed above plot-
LABEL   -50 characters printed below plot-
```

Every \* PLOT sequence defines a plot of swept frequency vs. log polar transfer function at a given node for a given order. As the circuit analyses proceed through the iterations of the frequency sweep, each frequency value (together with its associated nonlinear transfer function) at a particular node and order define one point on a curve. The total number of points on any given curve is equal to the number of steps in the frequency sweep selected for plotting. For example:

```
* GENERATOR
NODE    1      0
FR      1      0.1E6      1.0E6      10      LIN
FR      2      2.0E6
IMP     50.    0.
* PLOT
FR      1      LIN
NODE    6
ORDER   2
```

The curve resulting from this sequence will consist of ten points where the x-values are the frequency values .1 MHz, .2 MHz, ... 1.0 MHz, and the y-values are the second order log polar transfer functions at node 6.

If several plots are desired, for several nodes or at both 1st and n'th orders, then each plot must be defined by separate \*PLOT card sequence. For example, three plots could be obtained by the following series of \* PLOT specifications:

```
* PLOT
FR      1      LIN
NODE    6
ORDER   1
LABEL   Node 6 - First Order
* PLOT
FR      1      LIN
NODE    6
ORDER   3
LABEL   Node 6 - Third Order
* PLOT
FR      1      LIN
NODE    10
ORDER   3
LABEL   Node 10 - Third Order
```

A present program limitation restricts the total number of plot definitions in a single NCAP input deck to ten. The \* PLOT card sequence(s) may be placed anywhere in the input deck.

Plotted output occurs after all circuit analyses have been performed and the standard printed output is completed. For each plot specified in the input deck, two graphs result: one for the magnitude of the transfer function and one for the angle. In addition to the two graphs, the x - y coordinates of each point are tabulated and printed before the plot is output.

When plotted output is specified for circuits which involve linear component sweeping or device modification, several curves will result for each plot. The number of curves generated is dependent upon the maximum number of steps in linear component sweeps and the number of times the circuit undergoes

ication. In such instances the analyst must be aware of the  
 ence in which the operations will be performed in order to  
 understand not only how many curves will result, but also to  
 enable him to correctly relate a particular curve on a plot to  
 its exact circuit definition. For example, consider the  
 following NCAP input specifications:

```

* START CIRCUIT
* GENERATOR
NODE      1      0
FR        1      2.0E6
FR        2      0.1E6      1.0E6      10      LIN
IMP       50.     0.
* LINEAR COMPONENTS
C         1      2      6.6E-9
R         2      0      20.2E3
R         5      0      .01
C         4      0      4.E-12
R         4      0      1.98E3
C         4      6      8.1E-9
R         6      0      10.E3      20.E3  2      LIN
* MODIFY
R         6      0      10.E3      20.E3  2      LIN
* TRANSISTOR
NODE 2
  4.6      10.0      140.      .348
  .01      .150      .125      8.2
  21.2E-12  1.0      340.E-12  59.1E-9
  10.1     1.0E5     0.0      1.5E-12
* MODIFY
  5.0      8.76      200.      .209
  .0043    .150      .125      8.2
  0.0      1.0      340.E-12  2.8E-9
  400.     1.0E5     0.0      1.5E-12
* PLOT
FR        2      LIN
ORDER    1
NODE     6
* END CIRCUIT
* END
  
```

For purposes of plotting, although this NCAP run would perform a  
 total of 40 circuit analyses, it defines four distinct circuits  
 and subsequently produces four curves of ten points each. A

summary of circuit iterations with pertinent element values follows:

<u>ITERATION</u>	<u>FR 2</u>	<u>R(6,0)</u>	<u>TRANSISTOR</u>
1	.1E6	10.E3	Original
2	.2E6	10.E3	Original
.	.	.	.
.	.	.	.
10	1.0E6	10.E3	Original
(End Curve no. 1)			
11	.1E6	20.E3	Original
12	.2E6	20.E3	Original
.	.	.	.
.	.	.	.
20	1.0E6	20.E3	Original
(End Curve no. 2)			
21	.1E6	10.E3	Modified
22	.2E6	10.E3	Modified
.	.	.	.
.	.	.	.
30	1.0E6	10.E3	Modified
(End Curve no. 3)			
31	.1E6	20.E3	Modified
32	.2E6	20.E3	Modified
.	.	.	.
.	.	.	.
40	1.0E6	20.E3	Modified
(End Curve no. 4)			

Every set of ten iterations constitutes a complete pass through the range of the frequency sweep and therefore produces one curve of ten points. The four circuit definitions are the result of a linear component sweep together with modifications. The first two circuits (hence curves) are produced by the two-step resistor sweep in the original \* LINEAR COMPONENTS definition. They use the original transistor definition together with R(6,0) first at 10K and then at 20K. The second two circuits are defined by the modified resistor, which reinstates the sweep, and a modified

transistor. The second pair of circuits use the modified transistor together with R(6,0) first at 10K and then at 20K.

When several curves are associated with a plot specification, as in the preceding example, the curves are all printed on a single set of axes. Due to a present program limitation, a maximum of five curves can be displayed on a single set of axes. If more than five curves are generated by an NCAP run, they are output in groups of five. The plotting characters used for multiple curves are \*, +, 0, x, and =.

**MISSION**  
**of**  
**Rome Air Development Center**

*RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C<sup>3</sup>I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.*