COMPUTER-AIDED DESIGN:
MONTE-CARLO DC ANALYSIS USING WATAND

BY

JANQ-FANG (JEFF) SUEN

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Electrical Engineering

Program

Philip C. Munro 7-August-1987
Advisor Date

Sally M. Hotchkiss August 17, 1987
Dean of the Graduate School Date

YOUNGSTOWN STATE UNIVERSITY

AUGUST, 1987

ABSTRACT

COMPUTER-AIDED DESIGN:

MONTE-CARLO DC ANALYSIS USING WATAND

Janq-Fang (Jeff) Suen
Master of Science, Electrical Engineering
Youngstown State University, 1987

A Monte-Carlo DC analysis and display post-processor based on the WATAND macro facility are designed to meet the requirements of flexibility, speed, and minimum storage for consistency with the WATAND package. The concepts of Monte-Carlo simulation, probability, and random numbers are employed. Gaussian distribution is used as a default distribution to generate random element values, but any user-defined distribution(s) may be used. The Monte-Carlo DC analysis provides statistical DC analysis for circuits with linear and nonlinear elements. Initial, maximum, minimum, mean and standard deviation values of changed elements, parameters, and specified outputs are displayed. A Monte-Carlo post-processor is designed to redisplay the results of an analysis and to provide selective display of a previous analysis to save computer time. Examples using a difference amplifier, a class A amplifier, and a power distribution system are provided. Several probability distributions are used in these examples.

ACKNOWLEDGEMENTS

First I wish to gratefully thank my thesis advisor, Dr. Philip Munro, for his ongoing guidance and noteworthy amount of time and energy to this thesis, and his family for the encouragement and passionate concern about me through the Bible study every Friday evening.

I would like to express my sincere gratitude to Dr. Hojjat Mehri in the Department of Industrial Engineering, Dr. Salvatore Pansino and Dr. Jalal Jalali in the Department of Electrical Engineering for their valuable comments on probability that are so helpful for the completion of the thesis. Also, my special thanks go to Prof. Robert Foulkes, who provided me with many a valuable practical technique when I worked as his laboratory teaching assistant.

I wish to gratefully acknowledge the assistance of the instructor in the English Department, Mrs. Joy DeSalvo, who carefully corrected my writing. I must also thank Mr. Carmine Schiavone at the reading lab for his immeasurable reviews. A special note of thanks is due to my American friend family, Dr. Charles Gebelein and his family, who provided me with a sweet home in Youngstown. I owe a special note of appreciation to my parents, who gave support to my study in the U.S., and my wife, Sufen, whose love, patience and understanding encouraged the development and writing of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 OBJECTIVE

The complication in manufacturing integrated circuits (IC), and the rapid growth in very large integrated circuits (VLSI) have made it impossible to simply use the traditional "breadboard" method to test circuits. The employment of a computer simulation program to aid in the designing and testing of circuits has become one of the most efficient methods [1]. Computer-aided design (CAD) greatly assists the circuit designer in the jobs that would be impossible or difficult to achieve without a computer. The computer circuit simulation programs (e.g., SPICE: Simulator Program with Integrated Circuit Emphasis, WATAND: WATerloo ANalysis and Design) could be used to simulate circuit performances [1,2].

All manufactured elements, whether linear or nonlinear, will have different values from those labeled or declared by the manufacturers because of different temperature, material, and pressure during the manufacturing process. Usually the fault tolerance can be accepted if it does not exceed a certain limit. Because the variation of the element value or fault tolerance may severely affect the circuit output performances, the designers may want to know the statistical performances of a circuit. Therefore, a computer program that indicates the effects of the fault tolerance or the variations in a range of the elements is necessary.

A statistical method must be used to create statistical results for a circuit that will help the electrical designer to accurately adjust or evaluate a circuit more easily. The Monte-Carlo DC analysis (MCDC) and the Monte-Carlo display post-processor (MCDI) are designed for the statistical analysis of a circuit. The MCDC analysis is designed to execute the DC analysis and find the statistical input and output performances of a circuit. The MCDI post-processor is used to display the circuit statistical information which is created by MCDC analysis. This Monte-Carlo DC analysis will help the designers to test and evaluate a circuit.

1.2 MONTE-CARLO SIMULATION

The name of Monte-Carlo was generated from the association with gambling. The Monte-Carlo methods were first systematically developed in about 1944 and are used today to describe any kind of computational methods with random numbers [3]. The Monte-Carlo method has been applied to various areas, such as the evaluation of complicated integrals, the design of nuclear reactors, etc. However, the real use of the Monte-Carlo method as a research tool stems from the work on the atomic bomb during the second world war. This work involved a direct simulation of the probabilistic problems concerned with random neutron diffusion in fissile material.

In general, there are two categories of Monte-Carlo methods [4,5]. The first category is the probabilistic type which solves problems by using random process. The second one, the deterministic type, is not intrinsically random, and it only uses random numbers to evaluate a fixed quantity, for example, the area of an integration of a curve. Both methods are usually defined as the Monte-Carlo simulation (also called the stochastic-process simulation or the discrete-event simulation).

The probabilistic type is used to design the Monte-Carlo DC analysis in WATAND, because the probability distribution and random process are employed.

1.3 BASIC CONCEPT OF PROBABILITY

Statistics is a science that deals with the assembling , tabulation, analysis, and interpretation of quantitative and qualitative data [6]. It assists researchers in making an inference based on partial information and past experience.

In our actual daily lives, we cannot compile and examine all facts or figures that are involved. As a result, measurements are taken on a sample, not on the whole population. It is helpful to realize that accurate and precise results representing the population can be obtained.

A statistical approach is used for the Monte-Carlo DC analysis to solve a circuit problem, and it offers statistical results - mean, maximum, minimum, and standard-deviation values of a circuit to the designer. This process helps the designer to better understand the circuit performance and to improve the circuit before it is built in the laboratory or manufactured by the company.

1.4 OVERVIEW

This thesis is organized into three parts as a logical development of simulation and statistics concepts. The first part includes Chapters I through IV and covers the background knowledge of probability and random numbers. Chapters V and VI constitute the second part which describes the algorithm of the designed programs accomplishing Monte-Carlo DC analysis and display, and describes error messages. Application examples and conclusion make up the third part, consisting of Chapter VII and VIII.

An overview of each chapter is presented in the rest of this section. The Chapter II briefly introduces WATAND and its functions that are applied to the Monte-Carlo DC and display analyses. A probability discussion which describes the probability distribution and specific theories like Gaussian and exponential distributions is contained in Chapter III. Chapter IV provides the methods by which random numbers are obtained and the way that the seed number is decided on at the creation of a sequence of random numbers. Chapter V describes the implementation of the Monte-Carlo Analysis programs in WATAND. A description of the error and other messages from the WATAND environment comprises Chapter VI. Chapter VII contains three examples of the application of WATAND MCDC analysis. The first example uses a simple class A amplifier, the second example uses an IC difference amplifier, and the third one shows the flexibility of the Monte-Carlo DC analysis using a power distribution system. Finally, in Chapter VIII is the conclusion as a summary of the Monte-Carlo analysis and as a discussion of the future developments of Monte-Carlo analysis in WATAND.

CHAPTER II

WATAND PACKAGE

2.1 OVERVIEW

WATAND (WATerloo ANalysis and Design) is an interactive computer program for simulating linear and nonlinear electrical circuits [7]. Funded by the Natural Science and Engineering Research Council of Canada and developed by the Electrical Engineering Department of the University of Waterloo since 1972 [8], WATAND has been proven to be widely applicable and extremely convenient to the user. Version V1.10-00 was used for this work.

2.2 CIRCUIT DEFINITION IN WATAND

In entering into the WATAND environment the user must specify the circuit details by using a convenient non-formatted input. The available element types that can be employed are shown below [7]:

- linear and nonlinear resistors, inductors, and capacitors
- independent voltage and current source with waveforms
- linear and nonlinear controlled sources
- linear gyrators
- ideal operational amplifiers
- ideal impedance convertors
- ideal switches

- junction diodes
- bipolar transistors and FETs
- user-defined linear and non-linear multi-terminal elements

The user may create his own stand-alone building blocks, which can describe any circuit containing many repetitive sections.

2.3 <u>ANALYSES IN WATAND</u>

There are ten types of analyses and two types of post-processors which are built into WATAND. A brief introduction to these analyses is listed in Table 2-1.

2.4 <u>WATAND FACILITIES USED IN MONTE-CARLO ANALYSES</u>

In designing the Monte-Carlo DC and DI analyses, the DC analysis, wexec macros, #RUSER, and option commands are used and are discussed in the following paragraphs.

2.4.1 DC ANALYSIS

This WATAND analysis finds the dc operating point of a circuit. The DC analysis WATAND help file [9] is adapted and is listed in Table 2-2. In this DC analysis help menu, the menu shows how to use the DC analysis in WATAND. The user to use Monte-Carlo analysis in WATAND is assumed to have understood the DC analysis and the user's help menu in WATAND, so the DC analysis will not be discussed here.

Table 2-1 [9]

Analysis/Post-Processor Types

DC	DC Analysis. Finds dc operating points of linear and nonlinear circuits.
DT	DC Transfer Analysis. Finds dc transfer characteristics and/or multiple dc operating points.
TC	Transient Analysis. Finds time-domain response using companion model technique. Linear and nonlinear circuits.
TL	Transient Analysis. Finds time-domain response using inverse Laplace transform technique. Linear circuits only.
FR	Frequency Analysis. Uses AC Analysis for linear circuits, and small-signal analysis for nonlinear circuits.
FS	Sensitivity Analysis in the frequency domain. Uses small-signal analysis for nonlinear circuits.
SS	Steady-State Analysis. Extrapolates past transient to find the steady-state response.
PZ	Pole-Zero Analysis. Poles and zeros found for linear and nonlinear circuits.
SY	Symbolic Analysis. Symbolic form of transfer function polynomials wrt specified elements is displayed.
FO	Frequency-Domain Optimization performed wrt to circuit elements.
DF	Discrete Fourier Transform Post-Processor. Finds complex coefficients using results of a previous analysis.
DI	Display Post-Processor. Display results of a previous analysis.

Table 2-2

DC Analysis Help Menu

DC ANALYSIS

This WATAND analysis command finds the dc operating point of a circuit.

```
+-------+-------------------------------------------------------+
|  DC   |  PRint            OUtputs V p n       <options ...>   |
|       |  NOne                     V p                         |
|       |  CAll rname ...           V type.n                    |
|       |                           I type.n                    |
|       |                           G gname                     |
|       |                           ALl                         |
+-------+-------------------------------------------------------+
```

```
options:    (defaults listed first)

            ITeratio  200  | itmax
            SMultipl   10  | mult
            NIterati    0  | nimax
            NError  1D-12  | nemax
            NNolu       1  | nlu
            IPoint   ZERO  | ipname
            KEep      ALl          IN  DC      ON  A
                      OUtputs          fname       fmode
                      out-specs
                      NOne
```

where: (defaults)

PRINT NONE CALL	tells WATAND what to do with OUtput. See 'OUTPUT' helps in WATAND menu.
OUTPUTS	specifies which outputs to display. See 'OUTPUT' helps in WATAND menu.
ITERATIO	Maximum number of iterations (region boundaries crossed). (itmax=200)
SMULTIPL	Increasing this value may cure nonconvergence problems. (mult=10)
NITERATI	Maximum number of Newton-Raphson iterations to be performed. (nimax=0)
NERROR	A stopping criterion for the Newton-Raphson iterations. (nemax=1D-12)

NNOLU	The number of Newton-Raphson iterations without recomputing LU factors. (nlu=1)
IPOINT	specifies the initial-point name and is the initial guess. (ipname=ZERO)
KEEP	specifies keeping of the results of the analysis for post-processing. General form is 'KE what IN fname ON fmode'. (what=AL) (fname=DC) (fmode=A)

Notes:

1. The Katzenelson algorithm is used. If requested, the Newton-Raphson algorithm is then performed. (Enter NI...)
2. Independent sources are set to their values at t=0-.
3. For circuits with multiple solutions, the first-encountered solution starting from the initial point is taken. (DT analysis may be used to obtain multiple solutions.)

In the Monte-Carlo DC analysis, the initial point (IP) uses the first results of the DC analysis as the initial point (IP DC) to increase the execution speed of the DC analysis.

2.4.2 WEXEC MACROS

A wexec macro is most commonly used as a command in the WEXEC interactive environment, but it may also be called from the #Execute section of a WATAND source file [10]. It can be used to input a sequence of lines, but, it can also be used to check values, create and use variables, branch conditionally, etc.

2.4.3 REPEATING AN ANALYSIS WITH #RUSER

The #RUser command is used to repeat an analysis a number of times, and the #Repeat command is used to repeat an analysis for successive values of an element. In contrast to #Repeat, #RU does not alter an element's values, but calls a user subroutine before each analysis execution. For both #RU and #R, a keep file remains open during the repetitions if the analysis specifies the KEep option. Therefore, anything written to the keep file during repetitions is together in one file. The help menu to use the #RUser command is in the WATAND help menu listed in Table 2-3 [11].

2.4.4 #OPTION COMMANDS

Options are available in WATAND to control the operation, which affects the WATAND environment [7]. Two option commands are used in Monte-Carlo DC analysis, and these two commands are discussed below.

The first option, ANID ON|OFF, controls the printing of the analysis ID (including version number and date) at the beginning of output. The analysis name, current WATAND version number, current date, and table file name of an analysis is printed when ANID is set ON. The default condition is ON.

Table 2-3

#RUser Help Menu

```
+-----------------------------------------------------------+
|   #RUser nrep rname val1 val2 ... analysis-command ...    |
+-----------------------------------------------------------+
```

where:

nrep	is the number of times that the analysis is repeated.
rname	is the name of the subroutine that is called before each repetition of the analysis.
val...	are numbers which can be passed to the subroutine.
anal...	is the normal format of any analysis except PZ, SY, FO, DI, and user-defined analysis, which may be used.

The subroutine should be coded as follows:

```
SUBROUTINE rname(IREP,PAR)
REAL*8 PAR(1)
INTEGER IREP
  ...
RETURN
END
```

IREP	is the repetition number telling which repetition is about to be performed - 1st, 2nd, etc.
PAR	is an array of the values passed to the subroutine.

The second option is TIMESTAM ON|OFF. TIMESTAM controls whether or not the execution time of an analysis is printed. The default condition is ON. Because the Monte-Carlo statistical analysis does not want the DC analysis ID information and execution time to be printed for each DC analysis, the ANID and TIMESTAM parameters are set OFF in the beginning of Monte-Carlo DC analysis and ON in the end of the MCDC analysis.

2.4.5 FORTRAN AND ASSEMBLER SUBROUTINES

There are several subroutines written in Fortran and Assembler in the WATAND package that are used directly in the Monte-Carlo DC and DI analyses (e.g., PACKCH FORTRAN). Using these subroutines saves memory because all these subroutines are loaded in memory already and WATAND allows such use.

CHAPTER III

PROBABILITY

For designing the Monte-Carlo statistical analysis, the statistical results - mean value, standard deviation, etc. - are calculated; therefore the Gaussian and statistical results will be discussed in this chapter. The exponential distribution, which is used in an example as a user's distribution in Chapter VII, will be also discussed in this chapter. The function of cumulative distribution, which is used to generate the random numbers from the user-defined distribution by using the inverse transformation method, is discussed in section 3.1 in this chapter.

3.1 MEAN VALUE, STANDARD DEVIATION AND CUMULATIVE DISTRIBUTION [6,12]

When the random variables are used for getting the statistical results of a simulation, the output data should include at least the statistical mean value, i.e., arithmetic mean. The statistical mean value of the performance criterion, $\overline{Y}$, can be expressed as

$$\overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i \qquad (3-1)$$

where the Y_i's represent individual values of the performance criterion, and n is the number of samples.

The corresponding standard deviation, S, is formulated as

$$S^2 = \frac{1}{n} \sum_{i=1}^{n} [Y_i - \overline{Y}]^2 \tag{3-2}$$

or equivalently as

$$S^2 = [\frac{1}{n} \sum_{i=1}^{n} Y_i^2] - (\overline{Y})^2 \tag{3-3}$$

Equation 3-3 is used in designing the Monte-Carlo DC analysis for the standard deviation calculation.

Finally, the concept of the cumulative distribution will be discussed. Before going further, the idea of relative frequency has to be addressed. If the interval between maximum and minimum of Y_i is divided into m equal parts, then the relative frequency of those values of Y_i falling in interval j is

$$f_j = n_j / n \tag{3-4}$$

where n is the total number of Y_i's and n_j is the number of elements Y_i that fall into the jth interval and n is expressed as

$$n = \sum_{j=1}^{m} n_j \tag{3-5}$$

If Y_j is used to represent cumulative frequency of the observations which fall in intervals $j_1, j_2, \ldots, j_m$ then a cumulative distribution can be obtained from the above equations extended as

$$\begin{aligned} Y_1 &= f_1 \\ Y_2 &= f_1 + f_2 \\ &\;\vdots \\ Y_j &= f_1 + f_2 + \ldots\ldots + f_j \qquad (3\text{-}6) \\ &\;\vdots \\ Y_m &= \sum_{j=1}^{m} f_j \end{aligned}$$

If the user-defined distribution can be integrated then the cumulative distribution is used to generate non-uniform random numbers by employing the inverse transformation method. The researcher can use this approach to generate the random numbers from the user-defined distribution, and use these random numbers for the element error tolerance in Monte-Carlo DC analysis. An example will be discussed in Chapter IV.

3.2 GAUSSIAN AND EXPONENTIAL DISTRIBUTIONS

The Gaussian distribution is used as the default distribution in the MCDC analysis simulation program, and the Exponential distribution is used in the special user defined case described in Chapter VIII, so these two distributions are discussed below.

3.2.1 THE GAUSSIAN DISTRIBUTION [12]

Also called normal distribution, the Gaussian distribution is characterized by a symmetrical bell shaped probability density function, given by

$$f(x|U,S^2) = 1/(S\sqrt{2\pi})\exp\{(-1/2)*[(x-U)/S]^2\} \quad -\infty<x<\infty \quad (3-7)$$

where U is the mean and S the standard deviation. The unit normal distribution (i.e., S=1 and U=0) of the probability density function is formulated as

$$f(z|0,1) = (1/\sqrt{2\pi})\exp(-z^2/2) \quad (3-8)$$

From the above equation, an easy way to obtain the unit normal random variates can be derived from the following formula

$$Z = (-2*\ln U_1)^{1/2}\sin(2U_2) \quad (3-9)$$

or

$$Z = (-2*\ln U_1)^{1/2}\cos(2U_2) \quad (3-10)$$

where U_1 and U_2 are (0,1) random numbers generated from a uniform distribution, and Z is the random number which will fit into the Gaussian distribution.

3.2.2 THE EXPONENTIAL DISTRIBUTION

The exponential distribution is used in many simulation problems, especially in those that include a sequence of arrivals and departures, such as the simulation of a bank teller's window, an airport, etc., and is applied to the application examples in Chapter VII. The corresponding probability density distribution is shown as below [12].

$$f(x|\beta) = \begin{cases} \beta e^{-\beta x} & x>0 \\ 0 & x\leq 0 \end{cases} \qquad (3\text{-}11)$$

The mean value is expressed as

$$E(x) = 1/\beta \qquad (3\text{-}12)$$

The process to create a random number from the exponential distribution can be expressed as [12]

$$x = -\left(\frac{1}{\beta}\right)\ln U \qquad (3\text{-}13)$$

where U is generated from the (0,1) uniform random number generator, and x is the random number which will fit into the exponential distribution.

CHAPTER IV

RANDOM NUMBER GENERATION

4.1 THE CREATION AND TEST OF RANDOM NUMBERS

In a circuit simulation study, it is necessary to have random numbers generated by a computer to simulate the circuit output performances by the random process. Since random numbers are generated in a reproducible sequence by deterministic techniques, all these random numbers are not real random numbers and are called pseudo-random numbers.

A method adopted by the computer is the center-square method [12], in which the random number n_i is chosen by squaring the old random number and retaining the middle digits. For this reason if n_i has r digits, n_i^2 must have 2r or (2r-1) digits. This method is expressed as

$$n_{i+1} = \{ \text{ middle digits of } n_i^2 \} \tag{4-1}$$

Modern random number generators are based on the use of congruent numbers. The power residue method is the most commonly accepted one. It can be formulated as

$$n_i \equiv a n_{i-1} (\text{mod } m) \tag{4-2}$$

where the symbol '$\equiv$' signifies congruence, n_i and n_{i-1} are successive random integers, and a (the multiplier) and m (the modulus) are specified.

Pseudo-random numbers satisfy the criteria of randomness, large period, reproducibility, and computational efficiency. To assess the randomness of pseudorandom numbers, several test standards are necessary. The Chi-Square statistical test (x^2) is the most popular one and it can be expressed as

$$x^2 = (O_1-E_1)^2/E_1 + \;.....\; + (O_k-E_k)^2/E_k$$

$$= \sum_{i=1}^{k} (O_i-E_i)^2/E_i \qquad (4-3)$$

where k represents different classes, O_i the number of observed events, and E_i the expected number of events in class i.

In addition to the Chi-Square method, there are many well known methods which have been widely used for statistical tests, such as Frequency Test, Gap Test, and Increasing and Decreasing Runs [12,13].

4.2 IMSL RANDOM NUMBER ROUTINES

IMSL (International Mathematical Subroutines Library) is a group of more than 500 library routines written in Fortran [14]. Some of the routines are used to generate pseudo-random deviates.

The random numbers can be acquired by using a Fortran IMSL random number routine, but first two very important parameters have to be decided on. The first parameter is the DSEED number that is classified double precision for

Fortran in all IMSL generators. Using DSEED as an initial value in starting the random number generator, the DSEED number will be replaced by a new seed (DSEED) on output so that in making a series of calls to the generator, a series of random number sequences is generated to the user. The second parameter that is important is the amount of random numbers that the user expects the generator to produce.

The uniform generator GGUBS (or GGUBFS) in IMSL is used by most of the IMSL non-uniform random generators as the initial basic routine. In the MCDC analysis, the Gaussian random deviate generator (GGNQF) of IMSL is used for the production of non-uniform distribution random numbers.

4.3 GGNQF ROUTINE IN IMSL [14]

As a normal or Gaussian random deviate generator, GGNQF is used as a function routine. The initial value of seed number for GGNQF must be a double precision (REAL*8) Fortran variable and must be chosen in the range from 1.DO to 2147483647.DO. A new seed number is used in every subsequent call and the new seed number is generated by the computer automatically. The highest constraint of the seed number is 2^{32-1} or 2147483647 because of the constraint of computer size. The format of the GGNQF call is shown as follows:

$$R = GGNQF(DSEED) \qquad (4-4)$$

where DSEED is the seed number and R is the random number. Each execution of this function routine generates one random number R.

The random numbers generated by the GGNQF routine are (0,1) unit normal random numbers in which the standard deviation is equal to 1 and mean is equal to 0. Therefore, the GGNQF outputs have to be transformed to non-unit normal random deviates (U,S^2) by applying the following formula:

$$Y = R*S+M \tag{4-5}$$

where S is the standard deviation selected by the user, and M is the assigned mean value.

4.4 INITIAL NUMBER IN RANDOM NUMBER ROUTINE

Sometimes, one may want the random numbers created by a computer to look more like real random numbers. This can be achieved by choosing the initial seed number randomly.

If one does not assign a seed number in the Monte-Carlo DC analysis for the random number generated routine, the MCDC analysis will automatically pick the current CPU time, exercise a certain mathematical calculation, enlarge it and then make it double precision. For example, the seed number will be 23456789.DO in double precision when the CPU time is 0.123456789. Since the first digit after the decimal of the CPU time will change more slowly than the rest of the digits, the first digit after the decimal, e.g, the 1, is omitted to make the seed number more random. If the same initial value is used to create random numbers then the computer will generate the same sequences of random numbers.

4.5 RANDOM NUMBERS CREATED BY THE USER

The uniformly distributed random numbers are used with the inverse transformation method to generate a specified distribution [12]. Given a probability density function f(x), the user can easily integrate it into a cumulative distribution F(x) by employing the following formula.

$$F(x) = \int_{-\infty}^{x} f(x')dx' \tag{4-6}$$

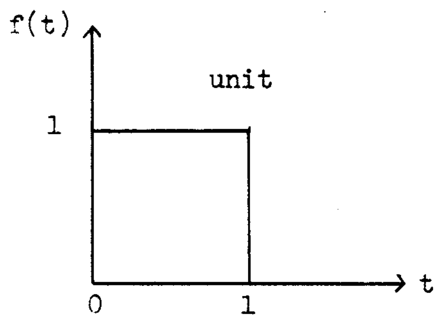
where F(x) is $0 \leq F(x) \leq 1$. Next, y is set to be equal to F(x), then

$$x = F^{-1}(y) \tag{4-7}$$

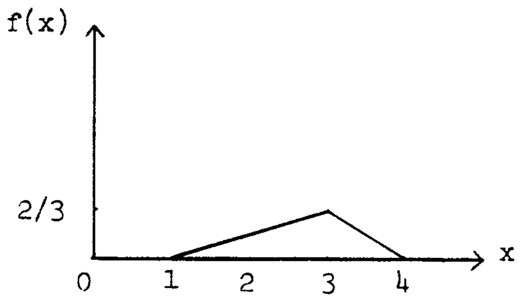
In the following example the inverse transformation method is used to acquire a definite formula for x as a function of U, which represents the uniform random numbers for the probability density function that follows. The uniform distribution is shown in Fig. 4-1a, and the user-defined distribution (triangular distribution) is shown in Fig. 4-1b.

The next approach shows how to generate user-defined random numbers by using the inverse transformation method from the user-defined distribution.

$$f(x) = \begin{cases} (x-1)/3 & 1 \leq x \leq 3 \\ (8-2x)/3 & 3 < x \leq 4 \\ 0 & x < 1 \text{ and } x > 4 \end{cases}$$



(a)



(b)

Fig. 4-1a. Uniform Distribution
b. Triangular Distribution

The first case is for $1 \leq x \leq 3$ and the cumulative distribution can be calculated as

$$F(x)=(1/6)*x^2-(1/3)*x+(1/6) \qquad (4\text{-}8)$$

then the formula listed below can be obtained from eq. 4-8.

$$x=1+(6U_i)^{1/2} \qquad 0 \leq U_i \leq 2/3 \qquad (4\text{-}9)$$

The second case is for $3<x\leq 4$ by using an initial value 2/3, which is the upper constraint of eq. 4-9. The cumulative distribution can be calculated as

$$F(x)=(8/3)*x-(1/3)*x^2-5+(2/3) \qquad (4\text{-}10)$$

then the formula listed below can be obtained from eq. 4-10

$$x=4-(3-3Ui)^{1/2} \qquad 2/3<U_i \leq 1 \qquad (4\text{-}11)$$

The Fortran program below is used to generate the user-defined random numbers for the triangular distribution. The equations 4-9 and 4-11 are used to generate the random numbers of the user-defined triangular distribution.

```
      REAL*8 DSEED
      REAL R,XC
      DSEED=12345670D0
      K=30
      WRITE(6,40)
      DO 30  I=1,K
      R=GGUBFS(DSEED)
      IF(R.GT.(2.0/3)) GOTO 10
      XC=1+(6*R)**(1.0/2)
      IF(XC.LT.1.0.OR.XC.GT.3.0) GOTO 999
      GOTO 20
10    XC=4-(3-3*R)**(1.0/2)
      IF(XC.LT.3.0.OR.XC.GT.4.0) GOTO 999
20    WRITE(6,50)R,XC
30    CONTINUE
40    FORMAT(' *****  UNIFORM  *****     *****   USER  DIST.
     ******')
50    FORMAT(2X,1PE16.5,10X,1PE16.5)
999   STOP
      END
```

Listed below are the computer outputs of the random numbers for the uniform distribution and the user-defined distribution. These outputs were run under the CMS environment.

```
*****  UNIFORM  *****      *****  USER DIS.  *****
     6.21772D-01               2.93148E+00
     1.24945D-01               1.86584E+00
     9.58064D-01               3.64530E+00
     1.74183D-01               2.02230E+00
     4.99902D-01               2.73188E+00
     8.57875D-01               3.34703E+00
     3.07061D-01               2.35734E+00
     7.69100D-01               3.16771E+00
     2.58645D-01               2.24574E+00
     4.46298D-02               1.51747E+00
     9.30042D-02               1.74701E+00
     1.22148D-01               1.85609E+00
     9.44539D-01               3.59210E+00
```

8.67776D-01	3.37018E+00
7.03920D-01	3.05754E+00
7.85852D-01	3.19847E+00
8.16521D-01	3.25809E+00
2.66561D-01	2.26466E+00
9.70551D-02	1.76311E+00
2.06466D-01	2.11301E+00
7.02845D-02	1.64939E+00
2.71808D-01	2.27705E+00
2.70418D-01	2.27378E+00
9.19866D-01	3.50969E+00
1.95145D-01	2.08207E+00
8.01582D-01	3.22847E+00
1.92879D-01	2.07577E+00
7.15284D-01	3.07580E+00
7.79894D-01	3.18740E+00
6.86117D-01	3.02961E+00

All of the thirty random numbers located in the above right hand column fit into the triangular distribution, and in the left column fit into the uniform distribution. The more random numbers are generated, the better the random numbers will fit into the triangular and uniform distributions.

CHAPTER V

IMPLEMENTATION OF MONTE-CARLO DC AND DI ANALYSES

5.1 OVERVIEW

There are one wexec macro and ten Fortran subroutines altogether in the MCDC analysis - one main control program (the MCDC wexec macro) and ten Fortran subroutines (MDINTL, MDREAD, MDPACK, MDCHCK, MDRUSR, MDRAND, MDIOUT, MDMODL, MDCHNG, MDSRCE). In the MCDI analysis, the main control macro, MCDI wexec, and four Fortran subroutines, MDINPT, MDIALL, MDCHCK, MDIOUT are used. All the Fortran subroutines mentioned above are listed in Appendix C.

There are three important rules that were followed in designing these programs - 1) to execute the Monte-Carlo analysis with a maximum speed, 2) to use a minimum computer memory, and 3) to allow the greatest flexibility. The functions of the subroutines and the procedures to complete all the subroutines and the two wexec macros are discussed in the following sections of this chapter.

Fortran IV is used with the FORTGI compiler in designing the Fortran subroutines for the MCDC analysis and MCDI post-processor, and all these Fortran subroutines are compatible with Fortran 77 or VS.

5.2 <u>PROGRAMS COMPRISING MCDC ANALYSIS</u>

In the process of designing the MCDC analysis, the speed of execution and the space saving of memory are the two important factors that must be balanced. If the speed of executing a program is of more importance, the program segments that demand considerable time may be designed in the form of a Fortran subroutine to accelerate the execution. In addition, to avoid reading data or writing outputs from the disk also increases execution speed.

5.2.1 MCDC WEXEC MACRO

The MCDC wexec macro is written to direct the entire development of the MCDC analysis, and this macro is shown in Table 5-1.

In the MCDC wexec macro, Lines 1 and 2 define the parameter names and assign default values for the parameters DSEED and NS. From Line 3 to Line 6, the parameter &ICODE is assigned a -1 and the subroutine MDCHCK is called to check to be sure the macro is executing in the WE environment. If the parameter &OUT(1) sent back from MDCHCK equals -1, then the macro is stopped because it is not executing in the WE environment. Line 7 checks to be sure that no positional parameter values are entered following the command word MCDC. Such values cause a branch to Line 36, where an error message is displayed and the execution of this macro is stopped. Lines 8 and 9 examine whether or not the parameters DSEED and NS are invalid (i.e., DSEED < 0 and

Table 5-1 MCDC Wexec Macro

```
Line    Program
01 &PARAM DSEED NS WRTL WRTM DC
02 &DEFAULT DSEED 0 NS 50
03 &ICODE = -1
04 &CALL MDCHCK OUT ICODE
05 &IF &OUT EM &GOTO -ERR10
06 &IF &OUT(1) EQ -1 &GOTO -ERR9
07 &IF &1 NM &GOTO -ERR1
08 &IF &DSEED LT 0 &GOTO -ERR2
09 &IF &NS LT 1 &GOTO -ERR3
10 &NSDC = &NS + 1
11 &IF &WRTL EM &IF &WRTM EM &GOTO -ERR4
12 &IF &DC EM &GOTO -ERR5
13 #E DC OU KE OU PR NO &DC
14 &ICODE = 1
15 &CALL MDCHCK OUT ICODE
16 &IF &OUT(1) EQ -1 &EXIT
17 &CALL MDINTL RSLT NSDC
18 &IF &RSLT(1) EQ -1 &GOTO -ERR6
19 &IF &RSLT(1) EQ -2 &GOTO -ERR7
20 &CALL MDREAD ELMOUT RSLT
21 &IF &ELMOUT EM &GOTO -ERR10
22 &IF &ELMOUT(1) EQ -1 &EXIT
23 &IF &ELMOUT(1) EQ -2 &GOTO -ERR8
24 #O ANID OFF TIMEST OFF
25 DC &DC PR NO KE NO
26 #E DC KE OU IN MCDC ON A &DC
27 #RU &NSDC MDRUSR &RSLT &NS &DSEED &ELMOUT DC IP DC NO
28 &ICODE = 1
29 &CALL MDIOUT RS ICODE NS RSLT
30 &IF &RS(1) EQ -1 &EXIT
31 &IF &RSLT(1) EQ 2 &MESSAGE # NOTICE # USE MCDI TO DISPLAY OUTPUTS
32 #E DC KE NO IP ZERO
33 &MESSAGE # NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
34 #O ANID ON TIMEST ON
35 &EXIT
36 -ERR1 &ERROR PARAMETER &1 NOT RECOGNIZED
37 &EXIT
38 -ERR2 &ERROR DSEED CANNOT BE LESS THAN ZERO
39 &EXIT
40 -ERR3 &ERROR NS MUST BE GREATER THAN ZERO
41 &EXIT
42 -ERR4 &ERROR MISSING WRTL/WRTM SPECIFICATION
43 &EXIT
44 -ERR5 &ERROR MISSING DC ANALYSIS OUTPUT SPECIFICATION
45 &EXIT
46 -ERR6 &ERROR DC ANALYSIS OUTPUT CANNOT BE 'ALL'
47 &EXIT
48 -ERR7 &ERROR MCDC ANALYSIS MUST USE 'KE OU/ALL' ONLY
49 &EXIT
50 -ERR8 &ERROR TOTAL NUMBER OF VARIED ELEMENTS CAN NOT EXCEED 20
51 &EXIT
52 -ERR9 &ERROR MCDC ANALYSIS VALID IN WEXEC ENVIRONMENT ONLY
53 &EXIT
54 -ERR10 &ERROR INSUFFICIENT STORAGE
55 &EXIT
56 #* 03-AUG-1987  BY J.F. SUEN  EE DEP.  YSU
```

NS < 1), and if so, it causes branches to Line 38 and 40, respectively, and an error message is displayed while the execution of the MCDC analysis terminates. Line 10 makes the value of &NSDC one more than the number of samples (NS) for the MDRUSR subroutine to execute one more time than the number of samples.

Line 11 checks whether both the parameters WRTL and WRTM are blank, and if so, it causes a branch to Line 42 where an error message is displayed and the MCDC analysis stops. Line 12 displays an error message and stops the execution of MCDC when learning that nothing is assigned to the parameter DC. Line 13 sets the DC analysis with the user's specification. The order of entry from left to right on the DC lines is important and the old specification is replaced by the user's specification once the latter is entered. Lines 14 through 16 check the error flag in the DC analysis by calling the MDCHCK subroutine. The error flag, which is in the WATAND memory, is set to zero when the DC analysis is ready to run; otherwise it set to 1. If the error flag is 1, the output parameter &OUT(1) is assigned a -1 to stop the MCDC analysis execution. Lines 17 through 19 check whether the parameters KE and OU are correctly specified in the DC analysis by calling the MDINTL subroutine. The conditions KE NONE and OU ALL are not allowed here. If the value of &RSLT(1) passed back from MDINTL is less than 0 because KE and OU are not correctly specified, such incorrect specifications of KE and OU cause branches to Line 46 and Line 48, respectively, and then an error message is displayed and the execution stops. Lines 20 through 23 obtain the element number, the element type, the two values following the element/parameter name and the index value by calling the MDREAD subroutine, and put all of them in the output parameter &ELMOUT if there are no errors with the element information. If there is an error, an error message is displayed by the MDREAD subroutine and &ELMOUT(1) is assigned a -1 to stop the execution.

Line 24 turns off ANID and TIMEST to prevent the title and the execution time from being displayed, for they do not need to be displayed at each execution of the DC analysis. Line 25 executes the DC analysis one time to generate the initial point for all the DC analyses to come, and thus the execution speed increases. Line 27 executes the #RUSER WATAND control command which calls MDRUSR before executing the DC analysis &NSDC times. MDRUSR replaces the elements/parameters with random values. The DC analysis output is stored in the disk after execution. Line 29 calls the MDIOUT subroutine to calculate and display the statistical outputs generated by the DC analysis. Line 32 resets the DC analysis to the default condition KE NO IP ZERO, and Line 33 displays this reset condition in a notice message. Line 34 resets ANID and TIMEST on to make the display of the title and execution time available. Error messages are presented on Lines 36 through 54.

5.2.2 MDINTL FORTRAN SUBROUTINE

On Line 17 in the MCDC macro, the MDINTL Fortran subroutine is called to check whether a correct specification of condition is present in the DC analysis while KE OU or KE ALL must be specified and OU ALL and KE NONE are not allowed. Then it counts and sends the number of elements and parameters in WRTL and WRTM to the MDREAD subroutine as input data.

Additionally the MDINTL Fortran subroutine is used to verify the input specifications of the DC analysis because KE NONE and OUTPUT ALL are not allowed. If an improper specification of condition is involved, MDINTL assigns an output code, -1, to display an error message and to stop the MCDC analysis from processing. Under normal circumstances when no error occurs, MDINTL delivers to the subroutine MDREAD the number of parameters following WRTL and WRTM and tells which parameter condition, KE OU or KE ALL, has been chosen in the DC analysis. If KE OU is used, the OU code is assigned a 1 and if KE ALL is used, the OU code is assigned a 2. The default condition is KE OU.

5.2.3 MDREAD FORTRAN SUBROUTINE

On Line 20 in the MCDC macro, the MDREAD Fortran subroutine is called to verify whether the element type, the element name, the two parameters following the element name, and the qualifier value of a linear or nonlinear element are correctly entered. The element number, the element type, the two parameters following the element name, and the qualifier value of all elements are all sent to the MDRUSR subroutine if all of them are correctly entered. On the contrary, if there is an error with any of them, an error message is displayed and the output code is assigned a -1 and then transmitted to the MCDC macro to stop the execution of the MCDC analysis.

During the verification of a modeled element, an END indicates the termination of the current model and the approach of a beginning of a new model when the current model is not the last model to prevent ambiguity between a parameter name and an element type.

5.2.4 MDPACK FORTRAN SUBROUTINE

The Fortran subroutine MDPACK is called by and loaded with the MDREAD Fortran subroutine in the same CMS file. The value of the subroutine argument TYPE determines the task that MDPACK is to do. If TYPE is assigned a 1, the element (or model) type number (ETP) is found and sent back to the calling subroutine MDREAD. For TYPE=2, the element number (ELN) is found and sent back to MDREAD, and for TYPE=3, the qualifier value is returned.

If the value of ETP or ELN is found to be equal to 0, the error code IER is assigned a 1; otherwise IER is assigned a 0. If the qualifier value is found valid, the parameter INDEX containing the position number of the qualifier value is passed back to the MDREAD subroutine and the error code IER is assigned a 0; otherwise IER is assigned a 1.

5.2.5 MDCHCK FORTRAN SUBROUTINE

Being an examining subroutine, the subroutine MDCHCK possesses two functions. First, MDCHCK is used to find out the environment, WI or WE, in which the WATAND circuits are. WE is the environment used if the variable WIWEFL in a WATAND common is equal to 2. The execution of the MCDC analysis will continue. Contrarily if WIWEFL is not equal to 2, the WATAND circuits are not in the environment of WE and MCDC analysis will cease. Secondly, MDCHCK is designed to verify the error flag for the DC analysis specifications. A 1 in the error flag, which is present in WATAND memory, means that there is no error in the DC analysis and the DC analysis is ready to run. On the contrary, if the error flag is equal to 0, there is an error in the DC analysis and the output code is assigned a -1 to stop the Monte-Carlo analysis from execution. Error messages are executed by WATAND itself when checking the specifications of the DC analysis.

In the subroutine of MDCHCK, the variable ATP is assigned two different values according to the analysis that the user wants to check. The ATP is assigned a 1 to allow MDCHCK to examine the DC analysis error flag, and is assigned a 12 for MDCHCK to verify the DI analysis flag.

5.2.6 MDRUSR FORTRAN SUBROUTINE

On Line 27 in the MCDC macro, the Fortran subroutine of MDRUSR is called to change the values of the elements for the DC analysis, which is executed after the MDRUSR Fortran subroutine. By calling the random number generator MDRAND, MDRUSR changes the element values, provides the DC analysis with the changed elements, and then stores the outputs in the disk. The analysis outputs can be stored with the user-assigned file name and the file mode or with the default name MCDC and the default mode A at the user's option; but the file types for the outputs are fixed to the special WATAND KEEP types, WKEEP and WKEEPID.

The number of executions of MDRUSR is one more than the number of samples (NS), because all elements whose values have been changed are reset to their original values in the last execution (NS+1). Additionally, in the last execution MDRUSR calculates the statistical values of all the changed elements, displays the statistical input values, and saves them on disk with the same file name and file mode of the DC analysis outputs mentioned in the above paragraph but with the special types, MDKPINFO and MDKPVALS. The output of the last execution of MDRUSR (NS+1) is not included in the statistical input results.

It must be noticed that if the user does not assign a DSEED number at the first execution, the current CPU time is used to generate the initial DSEED number.

5.2.7 MDRAND FORTRAN SUBROUTINE

The Fortran subroutine of MDRAND is used as a random number generator. In addition to the default subroutine which is designed by the author, there is an alternative - an MDRAND subroutine can be written by the user with user-defined distributions to meet non-Gaussian requirements of element error tolerances. Both situations are discussed in the following sections.

5.2.7.1 GAUSSIAN DISTRIBUTION

The Gaussian distribution is used as the element error tolerance distribution in the default MDRAND Fortran subroutine. The IMSL function subroutine GGNQF [14] is used as the normal random deviate generator to generate random numbers. The first parameter after the element name represents the error tolerance percentage, and if its value is equal to 0, then 10% is assigned as the default error tolerance for the element.

The changed value of the element, RANVL, is obtained from the following formulas:

$$STD = (INIVL * PAR1 / 100) * 2.0 / 6.0 \tag{5-1}$$

$$RANVL = INIVL + RAN * STD \tag{5-2}$$

where

STD = desired standard deviation of the random values,

INIVL = the initial value which is the desired mean,

PAR1 = the first parameter which is the percent tolerance, RAN = random number generated by the routine GGNQF,

RANVL = random value returned.

The above formulas are based on the assumption that the whole tolerance is from -3 to +3 sigma of the Gaussian distribution curve. Therefore, the range under this assumption includes 99.86% of the area in the Gaussian curve, but all of the generated random numbers are used in the default program even if they exceed +3 or -3 sigma.

Only one random number is generated for each call to MDRAND. The random number is then transmitted back to the subroutine MDRUSR.

5.2.7.2 USER DISTRIBUTION

That the user can use the user-defined distributions to design the MDRAND random number generator subroutine demonstrates a flexibility in the Monte-Carlo DC analysis for the user. To write this subroutine, the user should follow the format presented below:

```
      SUBROUTINE MDRAND(RANVL,TITLE,INPFL,IREP,ICOUNT,DSEED,
                        INIVL,PAR1,PAR2,IER)
      REAL*8    RANVL,DSEED,INIVL,PAR1,PAR2,TITLE(4)
      INTEGER   INPFL,ICOUNT,IER
      IF(IREP.NE.1) GOTO 10
      assign title name
  10     ...
         ...
      RANVL=random element value
      RETURN
      END
```

Where

RANVL is the random element variable returned,

TITLE	is the title displayed for the statistical outputs of MCDC,
INPFL	is the flag controlling the display of input information,
IREP	is the repetition number telling which repetition of the DC analysis is being performed,
ICOUNT	is the counting number showing which element or parameter is about to be changed,
DSEED	is the initial number for the random number generator,
INIVL	is the initial element value,
PAR1	is the first number following the element name,
PAR2	is the second number following the element name,
IER	is the error code returned.

RANVL is the random element variable to be transferred back to the MDRUSR subroutine to change the value of the element assigned by the user. Used to display a title for the statistical outputs of the MCDC analysis, TITLE, the second parameter, contains 32 bytes, and hence the information of TITLE can have a maximum of 32 characters. INPFL is an input flag. If this flag is set to an integer other than 1 by the user, the input data and their statistical information of the random variables of changed elements in MDRUSR will not be displayed.

Transferred from the subroutine MDRUSR, IREP is the repetition number which tells which repetition of the DC analysis is being performed, 1st, 2nd, etc. Also sent from

MDRUSR, ICOUNT is the counting number showing which element is about to be changed. DSEED is the seed number for the random number generator.

Transferred from the subroutine MDRUSR, INIVL represents the initial value of the element to be changed, and is used as the base value in reaching a new element value with a random number. Following the element or model parameter on the MCDC line, PAR1 and PAR2 are real numbers that the user can use to find a random value for the element. It is suggested that PAR1 be used as an error tolerance and PAR2 as a distribution-type code. IER is the error code sent back to MDRUSR. When an error occurs, the user can assign a number other than zero to IER to stop the DC analysis execution. The default value for IER is zero.

5.2.8 MDIOUT FORTRAN SUBROUTINE

On Line 29 in the MCDC macro, the Fortran subroutine MDIOUT is called to calculate statistical values in the MCDC analysis and to display them in the MCDI analysis. In MDIOUT, the outputs produced by the repeated DC analysis and the subroutine MDRUSR are loaded from the assigned disk and calculated to obtain necessary statistical information. The statistical values produced are then displayed and stored in the same disk where their original figures were derived from. The original data of these statistical values are not stored in the disk to save the memory in the disk.

MDIOUT is also used by the MCDI post-processor if DC KEEP OUTPUTS has been used in the MCDC analysis.

5.2.9 OTHER SUBROUTINES - MDMODL, MDCHNG AND MDSRCE

All the subroutines discussed previously in this chapter are called by the control macro MCDC. However, there are three other subroutines that are called by the MCDC Fortran subroutines. Their functions are discussed in this section.

MDMODL is called by MDREAD and MDRUSR to find the models that are to be changed, and to pass all the parameter information of these models to the MDCHNG subroutine along with code TYPE whose value is sent from the calling subroutine.

The Fortran subroutine MDCHNG is controlled by the subroutine MDMODL with TYPE as a control code. When TYPE is 1, the initial values of the model parameters are to be found in the WATAND circuit file. When TYPE is 2, the random values are obtained from the subroutine MDRAND, and used to change the values of the model parameters by calling the WATAND altering Fortran subroutine AMSECT. When TYPE is 3, MDCHNG verifies the existence of the model parameters assigned by the user on the MCDI input lines, and then finds the qualifier values for the parameters to return them to MDMODL in the argument, INDEX. When TYPE is 4, the model parameter names are found and passed back to MDMODL in the argument PNAME.

If there is no error in MDCHNG, IER is assigned a 0. For IER=1, the qualifier value is missing or invalid, and for IER=2, the model parameter is not found.

The Fortran subroutine MDSRCE performs certain jobs in regard to the value of the code TYPE transferred from its calling subroutine. When TYPE is 1, the original value will be found. When TYPE is 2, the random value will be used to change the voltage or current source value. When TYPE is 3, MDSRCE is used to check whether the qualifier value is correct. If the qualifier value is correct, the index number is assigned the qualifier value and transferred back to MDREAD; otherwise the error code IER is assigned a 1.

5.3 THE POST-PROCESSOR MCDI

As a post-processor, MCDI (Monte-Carlo DIsplay) is used to display the statistical input information of the MCDC analysis and its statistical outputs.

The choice of a parameter variable in the MCDC analysis greatly affects the way by which the MCDI displays. If KE OU has been selected in the MCDC analysis, then only the outputs that have been requested in MCDC are displayed by the MCDI analysis. On the contrary, if KE ALL has been picked, the node voltages and auxiliary variables are the exclusive outputs that can be displayed in MCDI, and because of the limitation of seven outputs as the maximum per analysis, the user can complete displaying the output by means of doing the MCDI analysis as many times as needed.

5.3.1 MCDI WEXEC MACRO

The MCDI wexec macro is used as the control macro of the MCDI analysis to display the input data and the statistical results of the MCDC analysis.

The parameter INPUT has to be specified ON or OFF with no default condition at the first execution of MCDI and at the time when the user wants to change its status. Indicating the user's desire of displaying or not displaying the input statistical information of MCDC, INPUT ON|OFF is the signal for the action of MCDI according to the user's wish.

The user must also specify the parameter variable OU to display required outputs of MCDC. As mentioned in the preceding sections, only outputs that are node voltages or auxiliary variables can be specified after OU if KE ALL has been used in MCDC.

The parameters USE and FM are used to specify the file name and the file mode, respectively, for MCDI to display the statistical results of the MCDC analysis. The default file name is MCDC and the default file mode is A.

Error messages are displayed if the above-mentioned parameters are not correctly specified, whereas notice messages are displayed as a status instruction.

In the MCDI wexec macro shown in Table 5-2, Lines 1 and 2 define the parameters and assign the default parameter values. Lines 3 through 6 check to be sure that the MCDI macro is in the WE environment so that the macro can continue executing. If the MCDI macro is not in WE, its execution stops and a branch goes to Line 42 where an error message is displayed. Also from Line 3 to Line 6, the capacity of WATAND memory is checked. Insufficient WATAND

Table 5-2 MCDI Wexec Macro

```
Line    Program
01 &PARAM INPUT OU USE FM
02 &DEFAULT USE MCDC FM A
03 &ICODE = -1
04 &CALL MDCHCK OUT ICODE
05 &IF &OUT EM &GOTO -ERR4
06 &IF &OUT(1) EQ -1 &GOTO -ERR5
07 &IF &1 NM &GOTO -ERR1
08 &PCODE = 0
09 &IF &OU NM &PCODE = 1
10 &IF &INPUT EQ ON &CODE = 1
11 &IF &INPUT EQ OFF &CODE = 0
12 &IF &CODE EM &GOTO -ERR2
13 &PASS INPUT
14 #E DI NO US &USE ON &FM
15 &ICODE = 12
16 &CALL MDCHCK OUT ICODE
17 &IF &OUT(1) EQ -1 &EXIT
18 &CALL MDINPT RSUL CODE PCODE
19 &IF &RSUL EM &GOTO -ERR4
20 &IF &RSUL(1) EQ -1 &EXIT
21 &IF &RSUL(2) EQ 1 &GOTO -NEXT
22 &IF &PCODE EQ 0 &GOTO -ERR3
23 #E DI OU &OU US &USE ON &FM
24 &CALL MDCHCK OUT ICODE
25 &IF &OUT(1) EQ -1 &EXIT
26 &CALL MDIALL RS RSUL CODE
27 &IF &RS(1) EQ -1 &GOTO -ERR3
28 &IF &RS(1) NE 0 &GOTO -WAR1
29 &EXIT
30 -NEXT
31 &ICODE = 2
32 &CALL MDIOUT RS ICODE RSUL
33 &EXIT
34 -ERR1 &ERROR PARAMETER &1 NOT RECOGNIZED
35 &EXIT
36 -ERR2 &ERROR INPUT-DISPLAY SPECIFICATION MISSING OR INVALID
37 &EXIT
38 -ERR3 &ERROR OU SPECIFICATION MISSING OR INVALID
39 &EXIT
40 -ERR4 &ERROR INSUFFICIENT STORAGE
41 &EXIT
42 -ERR5 &ERROR MCDI ANALYSIS VALID IN WEXEC ENVIRONMENT ONLY
43 &EXIT
44 -WAR1 &WARNING ONLY &RS(1) OUTPUT SAMPLES USED
45 &EXIT
46 #* 03-AUG-1987  BY J.F. SUEN  EE DEP.  YSU
```

memory causes a branch to Line 40 where an error message is displayed. Line 7 checks to be sure that no positional parameter values are entered following the command word MCDI. Such values cause a branch to Line 34, and an error message is displayed while the execution of this macro is stopped. Lines 8 through 12 check the parameters INPUT and OU. If ON is assigned to INPUT, &CODE is assigned a 1 whereas &CODE is assigned a 2 if OFF is assigned to INPUT. A blank or anything other than ON or OFF assigned to INPUT causes the displaying of the error message in Line 36 and the macro is stopped from execution. For OU, if no output is assigned, &PCODE is assigned a 0; in addition, an error message in Line 38 is displayed and the execution of MCDI is stopped if KE ALL has been specified in the MCDC analysis. If there is an output assigned, then &PCODE is set to 1. Line 13 allows the user to set INPUT on or off without assigning a default value. The user must assign an ON or an OFF to INPUT at the first execution of the MCDI analysis in the WE environment.

Line 14 sets the DI analysis in WATAND for the subroutines MDINPT and MDIALL to read the data from disk. Line 15 sets &ICODE to 12. Line 16 checks to be sure that the DI post-processor is properly set in Line 14 by using the error flag in WATAND, which should be zero in this case. If the error flag is 1, the output parameter &OUT(1) is assigned a -1 to stop the MCDI macro execution. Line 18 calls the subroutine MDINPT to display the statistical input data of the MCDC analysis if &CODE is a 1. If &CODE is not equal to 1, the statistical input data of the MCDC analysis is not displayed. From Line 19 to Line 22, the outputs of the MDINPT subroutine are checked. A 1 in the parameter &RSUL(2) indicates that KE OU has been used in the MCDC analysis, and the program goes directly to Line 32 to call the subroutine MDIOUT in order to display the statistical outputs of the MCDC analysis. Line 26 calls the subroutine MDIALL to display the requested outputs, and only the node voltages and auxiliary variables are available. Lines 34 through 42 present the error messages. In Line 44, a warning message is displayed to tell the user that the number of samples to be used in MCDI is different from the one in MCDC.

5.3.2 MDINPT FORTRAN SUBROUTINE

On line 18 in the MCDI wexec macro, the Fortran subroutine MDINPT is called to fulfill the displaying or not displaying of the input statistical data of the MCDC analysis according to the command out of the parameter variable INPUT. If INPUT is set ON, MDINPT reads and displays the input statistical data of MCDC which is stored in the disk. If the INPUT parameter is set OFF, the reading and displaying are skipped, and MDINPT ends.

5.3.3 MDIALL FORTRAN SUBROUTINE

On Line 26 in the MCDI wexec macro, MDIALL is a Fortran subroutine called to read the node voltages or auxiliary variables from the disk and to calculate and display the statistical results of these node voltages and auxiliary variables - means, minimums, maximums and standard deviations, when KE ALL has been used in the MCDC analysis to store node voltages and auxiliary variables.

The specification of any other outputs than node voltages and auxiliary variables for OU (e.g., element voltage or currents) will cause the appearance of an error message, and the output code , -1, will stop the MCDI macro.

CHAPTER VI

MESSAGES

During the process of the MCDC analysis or the MCDI post-processor, there are three kinds of special MCDC and MCDI messages, error, warning and notice messages that are displayed. These messages are described in the following sections. In addition to MCDC/MCDI messages, WATAND system messages are used in some places. For example, an error message is displayed if there is an incorrect specification in the DC analysis.

6.1 ERROR MESSAGES

All the error messages resulting from the MCDC analysis or the MCDI display post-processor are displayed with the wrong specifications of parameters. The MCDC or MCDI error message helps the user to find and fix the input specification error. When a WATAND built-in error message is displayed, an input line is redisplayed with the pointer, '$', which points to the incorrect specification, but this is not true for the error messages of MCDC or MCDI. Since the macro line is not redisplayed, an error message of MCDC or MCDI is displayed with the incorrect specification, but without the pointer.

For the error messages which follow, user-defined subroutine generated error messages for Fortran subroutines are displayed with '**', and macro error messages are displayed with '*'. In addition, the prefix '## ERROR ##' is displayed with the user-defined subroutine generated error messages, and the prefix '# ERROR #' is displayed with the macro error messages. The messages below are listed alphabetically and include a brief explanation.

DC ANALYSIS OUTPUT CANNOT BE 'ALL' *

OU ALL cannot be specified in the DC analysis.

DSEED CANNOT BE LESS THAN ZERO *

The seed number must be equal to or greater than zero.

DUPLICATED ELEMENT/MODEL PARAMETER NAME FOR xxx **

This message shows that the changed element xxx or the model parameter of model xxx has been duplicated.

ELEMENT/MODEL xxx CANNOT BE ALTERED **

The element or model xxx can not be altered (e.g., SW or SC).

INPUT-DISPLAY SPECIFICATION MISSING OR INVALID *

The input action for the MCDI post-processor is not correctly assigned or is not present.

INPUT INFORMATION TOO LONG **

22 bytes is the maximum length for input data, and the input information is too long.

INSUFFICIENT STORAGE *

There is not enough memory in WATAND for MCDC or MCDI to execute.

INVALID WRTL/WRTM SPECIFICATION FOR ELEMENT/MODEL xxx **

The element or model xxx is in a wrong WRTL/WRTM specification.

MCDC ANALYSIS MUST USE 'KE OU/ALL' *

In the MCDC analysis only KE OU or KE ALL can be used.

MCDC/MCDI VALID IN WEXEC ENVIRONMENT ONLY *

MCDC or MCDI is valid in the WE environment only.

MISSING DC ANALYSIS OUTPUT SPECIFICATION *

The DC analysis output specification is missing.

MISSING PARAMETER NAME FOR MODEL xxx **

The parameter name for model xxx is missing.

MISSING WRTL/WRTM SPECIFICATION *

The specification, WRTL or WRTM, is missing.

MODEL/PARAMETER xxx NOT RECOGNIZED **

The model or parameter xxx is not recognized.

MODEL xxx NOT RECOGNIZED **

The model xxx is not recognized.

NS MUST BE GREATER THAN ZERO *

The number of sample must be greater than zero.

OU SPECIFICATION MISSING OR INVALID *

The user did not assign an output to OU in MCDI.

PARAMETER xxx NOT RECOGNIZED *

The positional parameter xxx following the command word, MCDC or MCDI, is not recognized.

PARAMETER NAME xxx NOT RECOGNIZED **

The parameter xxx is not recognized.

TOTAL NUMBER OF VARIED ELEMENTS CANNOT EXCEED 20 *

In the MCDC analysis the total number of varied linear and nonlinear elements cannot exceed 20.

TYPE xxx NOT RECOGNIZED **

The element type xxx is not recognized.

xxx V-VALUE MISSING OR INVALID **

The qualifier value of the element xxx or model parameter xxx is missing or invalid.

6.2 NOTICE MESSAGES

Another group of messages are the notice messages. These notice messages are used to notify the user of the situation of the MCDC analysis or the MCDI display post-processor. All these notices are listed and explained below.

MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO' *

This message notifies the user that these DC-analysis specifications are set or reset as indicated.

USE MCDI TO DISPLAY OUTPUTS *

The MCDC analysis does not display the outputs when KE ALL is used in the DC analysis. This message notifies the user to use the MCDI post-processor to display outputs.

6.3 WARNING MESSAGE

There is one warning message that could occur in the MCDI post-processor as listed and briefly described below.

ONLY xxx OUTPUT SAMPLES USED *

This warning message tells the user that the data in the disk were read xxx times in MCDI, and xxx is different from the number of samples in MCDC.

CHAPTER VII

APPLICATION EXAMPLES

All the examples shown in this chapter are run under the CMS environment (Release 4 SLU 413).

7.1 EXAMPLE WITH GAUSSIAN AND USER-DEFINED DISTRIBUTIONS

One example run in four cases with the applications of the default distribution and/or the user-defined distribution to the MCDC analysis is presented in this chapter using a simple class A amplifier (Fig. 7-1).

In the first case, the Gaussian distribution is used and the NS (number of samples) is 50. The second case is the same as the first one with the exception that NS is 500. In the third case the KE ALL parameter condition is used instead of the default parameter condition KE OU. In the fourth case the user-defined distributions (i.e., the triangular distribution and the uniform distribution) and the Gaussian distribution are used with the number of samples equal to 50 in order to compare the results of this example with those of the first one. All four cases use the same initial seed number (DSEED) for the random number generator subroutines and the same circuit.

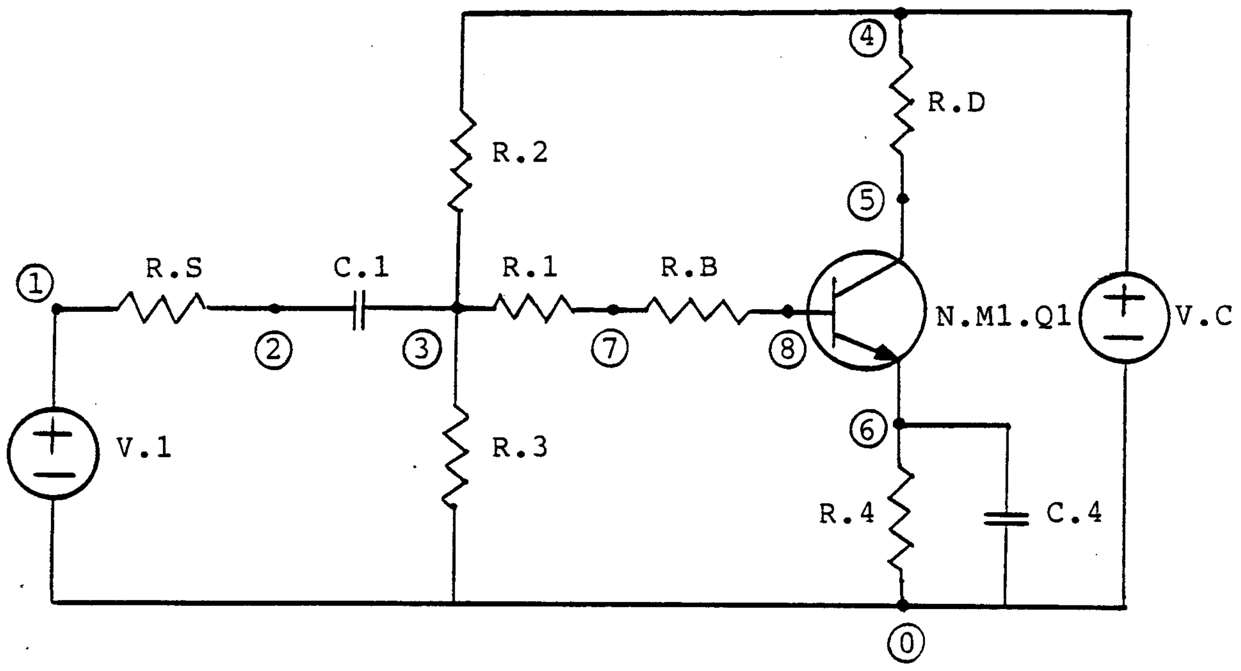


Fig. 7-1 Simple Class A Amplifier

The circuit file of a simple class A amplifier circuit (See EX9 [2] in the WATAND library) is listed below.

```
#T CLASS A TRANSISTOR CCT WITH PHILIPS BFR96 TRANSISTOR
#M
N.M1 IS 1.481E-15 GO .83E-4 BR 7.4 BF 70 .1 50 1E-4
* CE 6.774E-12 0 .7194 .3735 CC 2.817E-12 0 .4255 .2396
* FT 4E9 27E-1 .782 -10
* SA -100. -50. -20. -5. -1. 0 .3 .5 .6 .65 .7 .75 .8 .85 .9
*.95 1. 1.05  1.1 1.15 1.2
#D
V.1 1 0 SIN 1 1D6 0
V.C 4 0 DC 9
R.S 1 2 60
R.1 3 7 59
R.B 7 8 6.51
C.1 2 3 80MU
R.2 4 3 8.2K
R.3 3 0 2.7K
N.M1.Q1 8 6 5
```

```
R.D 4 5 330
R.4 6 0 220
C.4 6 0 7.5MU
#E
DC PRINT OU ALL I R.1 I R.D
FR GA BE 0.1 EN 1D13 LO 5 MA VB 0 50 PL OU V 5
TC EN 5D-6 DE 1D-8 VB 0 20 OU V 5 PL
#S
```

7.1.1 GAUSSIAN DISTRIBUTION WITH NS=50

This section uses the default Gaussian with NS equal to 50 to do the MCDC analysis. The following is the input command which is used to execute the MCDC analysis.

```
MCDC WRTL R.1 R.D V.C WRTM N.M1 BF 50 *
DC OU V 5 V R.4 I R.D I R.1 I R.4 NS 50 DSEED 1234570D0
```

MCDC is the leading word for the MCDC analysis. MCDC is followed by WRTL in which there are linear changed elements - resistors R.1 and R.D and the voltage source V.C. Since no number follows the linear element, the default error tolerance of 10% is used for all resistors and voltage source. Following WRTM is the model N.M1 and model parameter BF. The parameter BF has a 50% error tolerance. The "*" indicates condition on the next line where the DC analysis outputs are requested to display the voltages of node 5 and resistor R.4, and the currents of the resistors R.4, R.D, and R.1. Finally the number of samples NS is assigned 50, and the seed number DSEED is 1234570D0. The computer output response is listed below.

```
mcdc wrtl r.1 r.d v.c wrtm n.ml bf 50 *
dc ou v 5 v r.4 i r.d i r.1 i r.4 ns 50 dseed 1234570d0

MCDC V1.10-0e FC199501 26-JUL-87 17:49:15  YSUCMS    FILE: EX9

    NUMBER OF SAMPLES =   50          GAUSSIAN         DISTRIBUTION

INPUT                           INITIAL    MINIMUM    MAXIMUM     MEAN      ST.DEV

R.1                           5.900D+01  5.509D+01  6.553D+01  5.936D+01  2.019D+00
R.D                           3.300D+02  3.121D+02  3.547D+02  3.318D+02  9.802D+00
V.C                           9.000D+00  8.408D+00  9.691D+00  8.960D+00  3.205D-01
N.M1              BF          7.000D+01  4.420D+01  9.482D+01  7.249D+01  1.196D+01

OUTPUT               INITIAL       MINIMUM       MAXIMUM        MEAN         ST.DEV

V 5-0             7.07174D+00  6.59979D+00  7.50082D+00  7.03195D+00  2.31988D-01
V(R.4)            1.30442D+00  1.15043D+00  1.43909D+00  1.29709D+00  7.19923D-02
I(R.D)            5.84320D-03  5.13651D-03  6.44047D-03  5.81099D-03  3.27623D-04
I(R.1)            8.59936D-05  6.62503D-05  1.28565D-04  8.48813D-05  1.28017D-05
I(R.4)            5.92919D-03  5.22922D-03  6.54134D-03  5.89587D-03  3.27238D-04

MCDC  EXECUTION TIME=   0.467 SEC.

# NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
```

From the computer outputs above, the changed elements or model parameters are variated in the error tolerance 10% range. For example, the resistor R.D with the minimum value 312.1 ohms and maximum value 354.7 ohms shows that the random values are obviously within the interval 330$\pm$33 ohms. Also mean is close to the initial value while the standard deviation is about equal to (10%)*INITIAL*1/3, which shows that 3 sigma is about equal to the 10% tolerance.

7.1.2 GAUSSIAN DISTRIBUTION WITH NS=500

In this section, the number of sample NS is set to 500 to compare with the computer results in 7.1.1, and all other inputs and the circuit remain the same. By going through this case the user can examine the differences between the analyses with a different number of samples. The computer results are listed below:

```
mcdc wrtl r.1 r.d v.c wrtm n.ml bf 50 *
dc ou v 5 v r.4 i r.d i r.1 i r.4 ns 500 dseed 1234570d0

MCDC V1.10-0e FC199501 26-JUL-87 17:49:47  YSUCMS    FILE: EX9

    NUMBER OF SAMPLES =  500          GAUSSIAN         DISTRIBUTION

INPUT                      INITIAL    MINIMUM    MAXIMUM     MEAN      ST.DEV

R.1                       5.900D+01  5.400D+01  6.553D+01  5.908D+01 1.934D+00
R.D                       3.300D+02  2.979D+02  3.631D+02  3.307D+02 1.103D+01
V.C                       9.000D+00  8.045D+00  9.792D+00  8.997D+00 3.082D-01
N.M1          BF          7.000D+01  3.334D+01  1.038D+02  6.941D+01 1.172D+01

OUTPUT            INITIAL       MINIMUM       MAXIMUM        MEAN         ST.DEV

V 5-0          7.07174D+00  6.45946D+00  7.82270D+00  7.07350D+00  2.26722D-01
V(R.4)         1.30442D+00  1.08571D+00  1.46731D+00  1.29906D+00  6.85454D-02
I(R.D)         5.84320D-03  4.85312D-03  6.57778D-03  5.81652D-03  3.12027D-04
I(R.1)         8.59936D-05  6.31611D-05  1.68858D-04  8.83050D-05  1.33019D-05
I(R.4)         5.92919D-03  4.93507D-03  6.66960D-03  5.90483D-03  3.11570D-04

MCDC  EXECUTION TIME=   3.970 SEC.

# NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
```

From the display above, the error tolerance interval for resistor R.D is between 297.9 ohms and 363.1 ohms. These resistors exceed the limitation of 330±33 ohms which are the 10% of the error tolerance. Because 3 sigma is used to give an accuracy of 99.86% and there is a probability of 0.14% for the computer to generate a random number outside the error tolerance range, the greater the number of samples is, the more accurate the outputs for the statistical information should be. The problem with testing larger samples is that the CPU takes more time to calculate the results. We can see this execution time difference between a sample of 50 at 0.467 seconds and a sample of 500 at 3.97 seconds. The user has to decide between greater statistical accuracy and smaller computer execution time. With the number of samples equal to 500, the mean value is closer to the initial value and the standard deviation is closer to 1/3 of 10% tolerance than those in the NS=50 case.

7.1.3 GAUSSIAN DISTRIBUTION WITH NS=50 AND KEEP ALL

In the above two sections KE OU is used by default, but in this section KE ALL is employed, as the example shows how to use the MCDI post-processor. The same circuit and input data are used below for the MCDC analysis to compare with the case in 7.1.1. The computer results are

```
mcdc wrtl r.1 r.d v.c wrtm n.ml bf 50 *
dc ou v 5 v r.4 i r.d i r.1 i r.4 ke all *
ns 50 dseed 1234570d0

MCDC V1.10-0e FC199501 26-JUL-87 17:50:43  YSUCMS     FILE: EX9

    NUMBER OF SAMPLES =   50         GAUSSIAN        DISTRIBUTION

INPUT                      INITIAL    MINIMUM    MAXIMUM     MEAN       ST.DEV

R.1                       5.900D+01  5.509D+01  6.553D+01  5.936D+01  2.019D+00
R.D                       3.300D+02  3.121D+02  3.547D+02  3.318D+02  9.802D+00
V.C                       9.000D+00  8.408D+00  9.691D+00  8.960D+00  3.205D-01
N.M1          BF          7.000D+01  4.420D+01  9.482D+01  7.249D+01  1.196D+01

MCDC  EXECUTION TIME=   0.443 SEC.

# NOTICE # USE MCDI TO DISPLAY OUTPUTS
# NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
```

Of the computer results, only the statistical input data are displayed and the requested outputs of the DC analysis are ignored because KE ALL is used. The output node voltages and auxiliary variables must be displayed using the MCDI post-processor. On the last two lines of the computer results, the notice messages tell the user to use MCDI post-processor to display the outputs, and that the DC analysis is set to a default condition.

Only the node voltages and auxiliary variables can be requested as outputs for statistical information when KE ALL is used in the MCDC analysis. The MCDI post-processor in the following displays the node voltages 3, 5, 6, and 8, and auxiliary variable of current V.C. The computer results are listed below:

```
mcdi input on ou v 3 v 5 v 6 i v.c

MCDC V1.10-0e FC199501 26-JUL-87 17:50:43  YSUCMS     FILE: EX9

    NUMBER OF SAMPLES =   50          GAUSSIAN        DISTRIBUTION

INPUT                       INITIAL    MINIMUM    MAXIMUM     MEAN       ST.DEV

R.1                        5.900D+01  5.509D+01  6.553D+01  5.936D+01  2.019D+00
R.D                        3.300D+02  3.121D+02  3.547D+02  3.318D+02  9.802D+00
V.C                        9.000D+00  8.408D+00  9.691D+00  8.960D+00  3.205D-01
N.M1          BF           7.000D+01  4.420D+01  9.482D+01  7.249D+01  1.196D+01

OUTPUT               INITIAL       MINIMUM       MAXIMUM        MEAN          ST.DEV

V 3-0              2.05469D+00   1.89447D+00   2.19563D+00   2.04701D+00   7.50874D-02
V 5-0              7.07174D+00   6.59979D+00   7.50082D+00   7.03195D+00   2.31988D-01
V 6-0              1.30442D+00   1.15043D+00   1.43909D+00   1.29709D+00   7.19923D-02
I(V.C)            -6.69019D-03  -7.35454D-03  -5.93088D-03  -6.65403D-03   3.55047D-04

DISPLAY TIME=      0.030 SEC.
```

From the outputs above, the voltage of node 5 has the same outputs as those of the node voltage, 5, in 7.1.1.

Note that if the pass parameter INPUT is set OFF, then the input statistical information will not be displayed.

7.1.4 GAUSSIAN AND USER-DEFINED DISTRIBUTIONS WITH NS=50

This case shows how a distribution may be defined to fit an element's statistical requirements. There are three distributions, Gaussian, triangular, and uniform distributions, available in this case, and a new MDRAND Fortran subroutine is employed instead of the default MDRAND Fortran subroutine. The new MDRAND Fortran subroutine is listed below.

```
C***********************************************************
      SUBROUTINE MDRAND(RAN,TITLE,INPFL,IREP,ICOUNT,DSEED,
                        MEANVL,PAR1,PAR2,IER)
C***********************************************************
C
      REAL*8  DSEED,RAN,MEANVL,PAR1,PAR2,SDEV,TITLE(4),
     *        TPAR1,DABS,START,END,TCHANG(4),DSQRT
      REAL R
C
      DATA TCHANG/'GAUSSIAN',' TRIANGL','E & UNIF','ORM
     *     DIS.'/
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C...SET INITIAL VALUE
      TPAR1=DABS(PAR1)
      IF(PAR1.EQ.0D0) TPAR1=10D0
      START=MEANVL-MEANVL*TPAR1/100D0
      END=MEANVL+MEANVL*TPAR1/100D0
C
C...SET A NEW TITLE
      DO 10 I=1,4
10    TITLE(I)=TCHANG(I)
C
C...UNIFORM DISTRIBUTION
      IF(PAR2.NE.1D0) GOTO 20
      R=GGUBFS(DSEED)
      RAN=START+R*(END-START)
      GOTO 999
C
C...USER DEFINED TRIANGLE DISTRIBUTION
20    IF(PAR2.NE.2D0) GOTO 50
      R=GGUBFS(DSEED)
      IF(R.GT.(2.0/3)) GOTO 30
      XC=DSQRT(1+(6*R))
      GOTO 40
30    XC=DSQRT(4-(3-3*R))
40    RAN=(1.0/3.0)*(END-START)*(XC-1)+START
      GOTO 999
C
C...GAUSSIAN DISTRIBUTION
50    R=GGNQF(DSEED)
      SDEV=(MEANVL*TPAR1/100.0)*(2.0/6.0)
      RAN=MEANVL+R*SDEV
999   RETURN
      END
```

From the subroutine above, three distributions are available. If the second parameter following the element is a 1, the uniform distribution is used. If the second parameter is a 2, the triangle distribution, which is defined in Chapter IV, is employed. If nothing (default of 0) or another value is used for the second number, then the default Gaussian distribution will be used. The following are the input commands and the computer results.

```
mcdc wrtl r.1 r.d 10 2 v.c 10 1 wrtm n.ml bf 50 *
dc ou v 5 v r.4 i r.d i r.1 i r.4 ns 50 dseed 1234570d0

MCDC V1.10-0e FC199501 26-JUL-87 17:54:24  YSUCMS     FILE: EX9

    NUMBER OF SAMPLES =    50          GAUSSIAN TRIANGLE & UNIFORM DIS.

INPUT                          INITIAL    MINIMUM    MAXIMUM     MEAN       ST.DEV

R.1                          5.900D+01  5.509D+01  6.553D+01  5.936D+01  2.019D+00
R.D                          3.300D+02  3.092D+02  3.587D+02  3.360D+02  1.225D+01
V.C                          9.000D+00  8.144D+00  9.881D+00  8.942D+00  5.579D-01
N.M1           BF            7.000D+01  4.420D+01  9.482D+01  7.249D+01  1.196D+01

OUTPUT              INITIAL       MINIMUM       MAXIMUM        MEAN          ST.DEV

V 5-0             7.07174D+00  6.38551D+00  7.67114D+00  6.99517D+00  3.92834D-01
V(R.4)            1.30442D+00  1.09623D+00  1.49580D+00  1.29341D+00  1.20113D-01
I(R.D)            5.84320D-03  4.89448D-03  6.72341D-03  5.79446D-03  5.41146D-04
I(R.1)            8.59936D-05  6.26590D-05  1.35100D-04  8.46509D-05  1.43482D-05
I(R.4)            5.92919D-03  4.98286D-03  6.79909D-03  5.87912D-03  5.45968D-04

MCDC  EXECUTION TIME=   0.500 SEC.

# NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
```

From the computer results above, it is easy to compare the Gaussian distribution results with the Gaussian, triangular, and uniform distributions results since they have quite different input and output performances from the same circuit. For example, the uniform distribution which is used for the voltage source V.C has a mean value of 8.942V for the uniform distribution which is almost the same as the value 8.96V for the Gaussian distribution in section 7.1.1. These means should be equal for a large enough number of samples.

7.2 EXAMPLE USING IC DIFFERENCE AMPLIFIER

This example uses an IC difference amplifier circuit. The Gaussian distribution is employed as the error tolerance distribution for the element. The IC difference amplifier circuit is shown in Fig. 7-2 and its WATAND circuit file is shown below.

```
#T IC DIFFERENCE AMPLIFIER FOR MCDC ANALYSIS STUDY
#DE RDIV
MODEL RDIVDR 2 0 RVALS 20K .321
DATA  3
ELEMENTS
R  1 2  1 M
R  2 3  2 M
#M
RDIV.QE
N.QD  BF 1000 SA -15 -8 -.1 0 .2 ST .4 .9 .1  1.1 1.5 2.0
N.QE  BF 1000 SA -15 -8 -.1 0 .2 ST .4 .9 .1  1.1 1.5 2.0
#D
V.P  100 0  DC  15
V.N  200 0  DC -15
N.QD.Q1  10 15 20
N.QD.Q2  30 15 40
R.C1  100 20  750
R.C2  100 40  750
V.IN1  10 0  DC 0
```

```
V.IN2  30 0  DC 0
N.QE.Q3  50 60 15
R.E  60 200  200
RDIV.QE.RB  0 50 200
#E
DC PR OUT ALL I R.C1 I R.C2 I R.E
#GV CMODE ( V 20 + V 40 ) / 2
#S
```

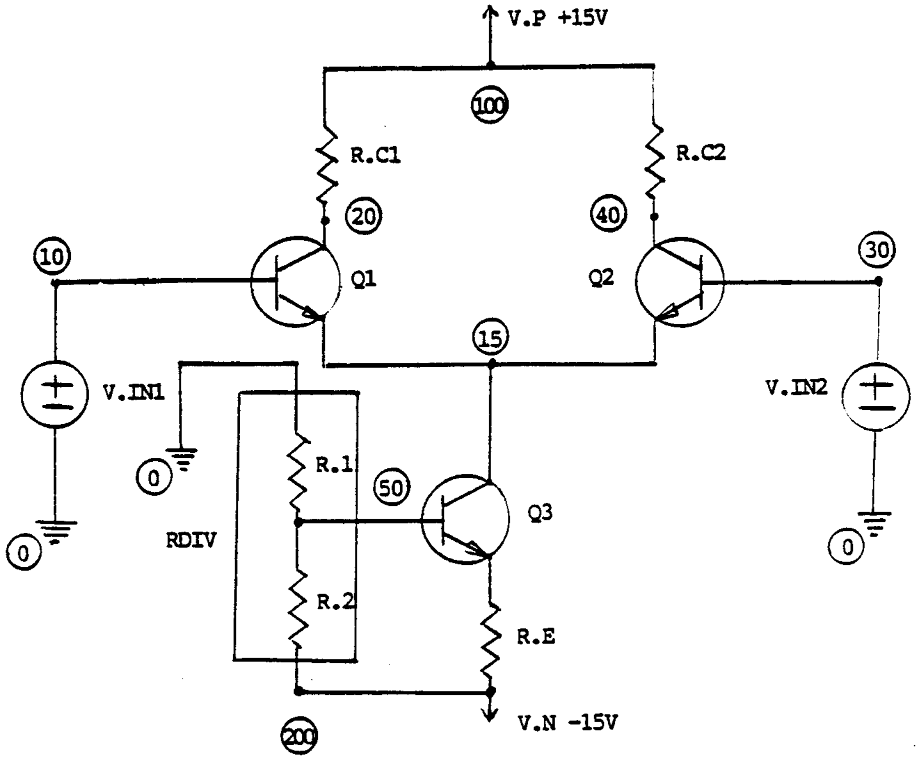


Fig. 7-2 IC Difference Amplifier

A user-defined model RDIV composed of the two resistors R.1 and R.2 is designed for the difference amplifier circuit. The total resistance value for the two resistors is 20K ohms and R.2 has a portion of 32.1% of the total resistor value, therefore the resistance value of R.2 is 6.42K ohms and the resistance value of R.1 is 13.58K ohms as subtracted by R.2 from the total value of 20K ohms. Listed below is the Fortran subroutine RDIVDR used by the user-defined model RDIV.

```
C**********************************************************
      SUBROUTINE RDIVDR(ICODE,PAR,VAR,VAL,EPAR,MIDA,DERIV)
C**********************************************************
C
C     #DE RDIV LINEAR RESISTANCE DIVIDER ELEMENT FOR MCDC
C
C     PAR(1) = R TOTAL
C     PAR(2) = K VALUE (SHOULD BE 0 < K < 1)
C     PAR(3) = R1
C     PAR(4) = R2
C
C     CREATED  25-JUL-1987  P.MUNRO/J.F.SUEN   EE DEP.   YSU
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8 PAR(1),VAR(1),VAL(1),EPAR(1),DERIV(1,1)
      INTEGER ICODE,MIDA(1)
      IF(ICODE.NE.1) RETURN
C
C...GET VALUES OF R2 AND R1 FROM R TOTAL AND K
      PAR(4)=PAR(1)*PAR(2)
      PAR(3)=PAR(1)-PAR(4)
      RETURN
      END
```

In the MCDC analysis of the difference amplifier circuit, the linear resistors R.C1, R.C2 and R.E are assigned an error tolerance percentage of 40%, and so is the parameter BF in both the transistors N.QD and N.QE. For the user-defined model RDIV, the first parameter RVAL, which represents the sum of the resistor values of R.1 and R.2, is assigned a 40% error tolerance, and the second parameter RVAL.V2, which represents the fraction of the total for the resistor R.2, is assigned a 5% error tolerance. Using two different models for Q1/Q2 and Q3 allows the parameters for Q1 and Q2 to be equal, while those of Q3 are statistically independent. The computer outputs of the MCDC analysis of the difference amplifier circuit are listed below:

```
#T IC DIFFERENCE AMPLIFIER FOR MCDC ANALYSIS STUDY
mcdc wrtl r.cl 40 r.c2 40 r.e 40 *
wrtm rdiv.qe rvals 40 rvals.v2 5 *
n.qd bf 40 n.qe bf 40 *
dc ou i r.cl i r.c2 v 20 v 40 v 20 40 g cmode

MCDC V1.10-0e FC199501 27-JUL-87 14:31:54  YSUCMS    FILE: DIFFAMP

    NUMBER OF SAMPLES =   50         GAUSSIAN        DISTRIBUTION

INPUT                        INITIAL    MINIMUM    MAXIMUM     MEAN      ST.DEV

R.C1                        7.500D+02  5.012D+02  9.741D+02  7.555D+02  1.072D+02
R.C2                        7.500D+02  5.609D+02  1.004D+03  7.473D+02  1.015D+02
R.E                         2.000D+02  1.498D+02  2.567D+02  2.023D+02  2.832D+01
RDIV.QE        RVALS        2.000D+04  1.412D+04  2.601D+04  2.000D+04  2.815D+03
RDIV.QE        RVALS.V2     3.210D-01  3.112D-01  3.314D-01  3.210D-01  4.721D-03
N.QD           BF           1.000D+03  7.348D+02  1.307D+03  1.019D+03  1.205D+02
N.QE           BF           1.000D+03  8.053D+02  1.443D+03  1.002D+03  1.204D+02

OUTPUT               INITIAL       MINIMUM       MAXIMUM         MEAN        ST.DEV

I(R.C1)          1.00352D-02  7.86250D-03  1.35808D-02  1.01171D-02  1.44150D-03
I(R.C2)          1.00352D-02  7.86250D-03  1.35808D-02  1.01171D-02  1.44150D-03
V 20-0           7.47359D+00  4.43249D+00  1.01119D+01  7.39292D+00  1.34368D+00
V 40-0           7.47359D+00  2.97110D+00  1.04583D+01  7.43278D+00  1.55318D+00
V 20-40          0.0         -3.38094D+00  6.02445D+00 -3.98589D-02  1.78557D+00
GV CMODE         7.47359D+00  4.86285D+00  9.58524D+00  7.41285D+00  1.14536D+00

MCDC  EXECUTION TIME=    0.900 SEC.

# NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
```

In the listing above, notice the input display for first and second values of RDIV.QE. Also notice the statistically independent values for N.QD and N.QE. In the output display, both difference and common mode voltages are given. The common mode value is generated by using a WATAND feature called generalized output.

7.3 EXAMPLE WITH EXPONENTIAL DISTRIBUTION

In this section a more complicated example is offered to show that using the MDRAND Fortran subroutine may accomplish a special purpose design.

For easy explanation, a power distribution system is illustrated in Fig. 7-3. Assume all the resistances in

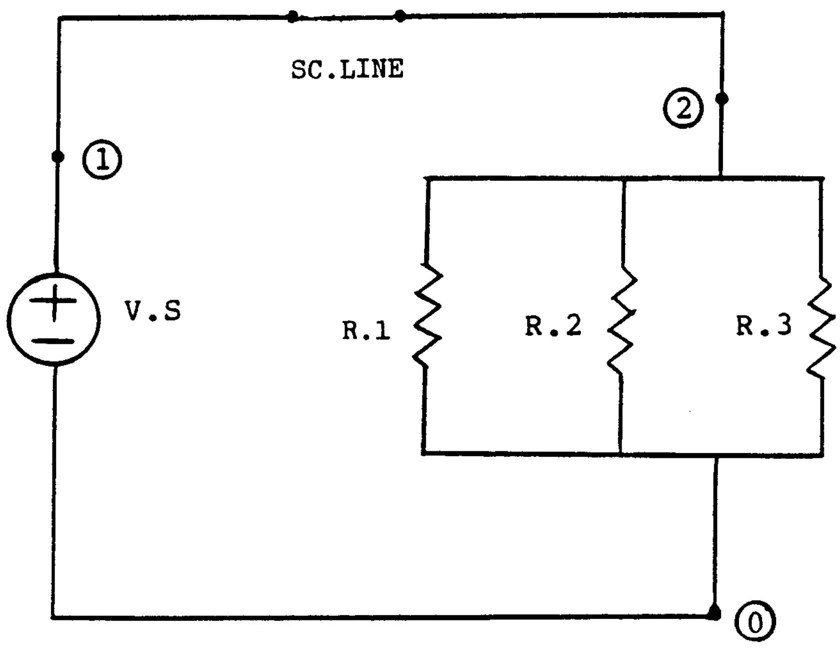


Fig. 7-3 Power Distribution System

area i are combined and represented by a single resistance R_i and the resistance values for each resistance are 10 ohms, 20 ohms, and 30 ohms respectively. Because all these power distribution system resistances have the chance to break down (i.e., the open circuit), the average breakdown for each of the three resistances is assumed to be once per 200 days. Assume furthermore that the time interval between breakdowns is an exponential distribution (Poission process), and it takes one day to complete fixing a breakdown. If a resistance breaks down, the resistance value will equal 1D+12 to make it like an open circuit in this resistance in the WATAND circuit, and other resistances will conform to a Gaussian distribution. Furthermore, for the error tolerance for each resistance it is assumed that R.1 equals 1.1 sigma, R.2 equals 1.2 sigma, and R.3 equals 1.3 sigma. By using Table 7-1 it is approved that the probability that the maximum current from a breakdown in the circuit will not exceed a specified interval is 99%. The WATAND circuit file and the Fortran subroutine MDRAND are listed below:

```
#T POWER DISTRIBUTION EXAMPLE
#D
SC.LINE   1   2
R.1       2   0   10
R.2       2   0   20
R.3       2   0   30
V.S       1   0   DC  120
#S
```

```
C************************************************************
      SUBROUTINE MDRAND(RAN,TITLE,INPFL,IREP,ICOUNT,DSEED,
                          INIVL,PAR1,PAR2,IER)
C************************************************************
C
      REAL*8   DSEED,PERIOD,RN,DLOG,TITLE(4),PRINT(3),
     *         RANVL,INIVL,PAR1,PAR2
C
      REAL     R,GGNQF
C
      INTEGER N(3),CLOCK(3),SMAL,P,NR,IPCODE,IREP,ICOUNT
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IER=0
      IF(IREP.GT.1) GOTO 20
```

```
C
C...SET INITIAL CONDITIONS
      IF(ICOUNT.NE.1) GOTO 50
      PERIOD=200
      INPFL=0
C
C...GENERATE FIRST EXPONENTIAL RANDOM NUMBER
      DO 10 I=1,3
      RN=GGUBFS(DSEED)
10    CLOCK(I)=-(PERIOD)*DLOG(RN)
C
C...WRITE THE TITLE, 1 IS THE WRITE UNIT
      WRITE(1,555)
      WRITE(1,666)
      WRITE(1,777)IREP,CLOCK
      GOTO 50
C
C...FIND NEXT EXPONENTIAL RANDOM NUMBER
20    IF(ICOUNT.NE.1) GOTO 100
      DO 30 I=1,3
      IF(N(I).EQ.0) GOTO 30
      RN=GGUBFS(DSEED)
      NR=-(PERIOD)*DLOG(RN)
      P=N(I)
      CLOCK(P)=CLOCK(P)+1+NR
30    CONTINUE
      WRITE(1,777)IREP,CLOCK
50    SMAL=CLOCK(1)
      DO 60 I=1,3
      IF(SMAL.LT.CLOCK(I)) GOTO 60
      SMAL=CLOCK(I)
60    CONTINUE
      DO 80 I=1,3
      IF(SMAL.NE.CLOCK(I)) GOTO 70
      N(I)=I
      GOTO 80
70    N(I)=0
80    CONTINUE
C
C...SET RESISTANCES VALUES
100   IF(N(ICOUNT).NE.0) GOTO 90
      R=GGNQF(DSEED)
      RANVL=INIVL+R*PAR1
      GOTO 110
90    RANVL=1D12
110   PRINT(ICOUNT)=RANVL
      IF(ICOUNT.EQ.3) WRITE(1,999) PRINT
      RETURN
555   FORMAT(/'*** EXPONENTIAL DISTRIBUTION TO BE USED ***')
666   FORMAT(/'** TIME **   * C #1 *   * C #2 *   * C #3 * '/)
777   FORMAT(I6,7X,I5,6X,I5,6X,I5)
999   FORMAT(' R VALUE   ',1P3E11.3/)
      END
```

TABLE 7-1 SELECT VALUES OF THE T DISTRIBUTION [12]

n-1	$t_{.995}$	$t_{.99}$	$t_{.975}$	$t_{.95}$	$t_{.90}$	$t_{.80}$	$t_{.75}$	$t_{.70}$	$t_{.60}$	$t_{.55}$
1	63.66	31.82	12.71	6.31	3.08	1.376	1.000	0.727	0.325	0.158
2	9.92	6.96	4.30	2.92	1.89	1.061	0.816	0.617	0.289	0.142
3	5.84	4.54	3.18	2.35	1.64	0.978	0.765	0.584	0.227	0.137
4	4.60	3.75	2.78	2.13	1.53	0.941	0.741	0.569	0.271	0.134
5	4.03	3.36	2.57	2.02	1.48	0.920	0.727	0.559	0.267	0.132
6	3.71	3.14	2.45	1.94	1.44	0.906	0.718	0.553	0.265	0.131
7	3.50	3.00	2.36	1.90	1.42	0.896	0.711	0.549	0.263	0.130
8	3.36	2.90	2.31	1.86	1.40	0.889	0.706	0.546	0.262	0.130
9	3.25	2.82	2.26	1.83	1.38	0.883	0.703	0.543	0.261	0.129
10	3.17	2.76	2.23	1.81	1.37	0.879	0.700	0.542	0.260	0.129
11	3.11	2.72	2.20	1.80	1.36	0.876	0.697	0.540	0.260	0.129
12	3.06	2.68	2.18	1.78	1.36	0.873	0.695	0.539	0.259	0.128
13	3.01	2.65	2.16	1.77	1.35	0.870	0.694	0.538	0.259	0.128
14	2.98	2.62	2.14	1.76	1.34	0.868	0.692	0.537	0.258	0.128
15	2.95	2.60	2.13	1.75	1.34	0.866	0.691	0.536	0.258	0.128
16	2.92	2.58	2.12	1.75	1.34	0.865	0.690	0.535	0.258	0.128
17	2.90	2.57	2.11	1.74	1.33	0.863	0.689	0.534	0.257	0.128
18	2.88	2.55	2.10	1.73	1.33	0.862	0.688	0.534	0.257	0.127
19	2.86	2.54	2.09	1.73	1.33	0.861	0.688	0.533	0.257	0.127
20	2.84	2.53	2.09	1.72	1.32	0.860	0.687	0.533	0.257	0.127
21	2.83	2.52	2.08	1.72	1.32	0.859	0.686	0.532	0.257	0.127
22	2.82	2.51	2.07	1.72	1.32	0.858	0.686	0.532	0.256	0.127
23	2.81	2.50	2.07	1.71	1.32	0.858	0.685	0.532	0.256	0.127
24	2.80	2.49	2.06	1.71	1.32	0.857	0.685	0.531	0.256	0.127
25	2.79	2.48	2.06	1.71	1.32	0.856	0.684	0.531	0.256	0.127
26	2.78	2.48	2.06	1.71	1.32	0.856	0.684	0.531	0.256	0.127
27	2.77	2.47	2.05	1.70	1.31	0.855	0.684	0.531	0.256	0.127
28	2.76	2.47	2.05	1.70	1.31	0.855	0.683	0.530	0.256	0.127
29	2.76	2.46	2.04	1.70	1.31	0.854	0.683	0.530	0.256	0.127
30	2.75	2.46	2.04	1.70	1.31	0.854	0.683	0.530	0.256	0.127
40	2.70	2.42	2.02	1.68	1.30	0.851	0.681	0.529	0.255	0.126
60	2.66	2.39	2.00	1.67	1.30	0.848	0.679	0.527	0.254	0.126
120	2.62	2.36	1.98	1.66	1.29	0.845	0.677	0.526	0.254	0.126
∞	2.58	2.33	1.96	1.65	1.28	0.842	0.674	0.524	0.253	0.126

From the MDRAND subroutine above, the INPFL is set to 0 to turn off the input information display. MDRAND generates the number of day(s) for resistance breakdown by using the exponential distribution with the smallest number of day(s) representing the soonest breakdown. After a breakdown happens to the resistance, a large value , 1D+12 ohms, is used for the resistance to show that it is an open circuit. The original breakdown day(s) plus one day to fix the breakdown and another number of day(s) generated by random number generator equals the period of time of the next breakdown. If the resistance does not break down, the resistance will be decided from the Gaussian distribution random number generator GGNQF. The computer results are listed below, and for each simulation, the first line shows the breakdown time for each resistance, and the second line shows the corresponding resistance value.

```
mcdc wrtl r.1 1.1 r.2 1.2 r.3 1.3 *
dc ou i r.1 i r.2 i r.3 i sc.line ns 50

*** EXPONENTIAL DISTRIBUTION TO BE USED ***

** TIME **    * C #1 *     * C #2 *     * C #3 *

    1            82           46           153
 R VALUE      1.101D+01    1.000D+12    2.910D+01

    2            82          223           153
 R VALUE      1.000D+12    1.935D+01    2.849D+01

                ...

   49          3636         3404          3340
 R VALUE      8.172D+00    2.220D+01    1.000D+12

   50          3636         3404          3384
 R VALUE      9.564D+00    2.140D+01    1.000D+12


OUTPUT             INITIAL       MINIMUM       MAXIMUM        MEAN          ST.DEV

I(R.1)           1.20000D+01   1.20000D-10   1.53090D+01   7.66201D+00   5.86808D+00
I(R.2)           6.00000D+00   1.20000D-10   6.68096D+00   4.02087D+00   2.76802D+00
I(R.3)           4.00000D+00   1.20000D-10   4.58392D+00   2.77109D+00   1.90680D+00
I(SC.LINE)       2.20000D+01   9.63805D+00   2.15499D+01   1.44540D+01   3.62550D+00

MCDC  EXECUTION TIME=    0.190 SEC.

# NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
```

Because the computer results are too long to be listed above, only part of them are shown as an explanation, and the detailed results are attached in Appendix A. The maximum current is an important concern to designers in this case because the maximum current may damage the circuit when a possible breakdown happens. Then the true mean value of the maximum current will fall in this interval [12]:

$$\overline{Y}-t_{n-1,1-\alpha/2}*(S/\sqrt{n-1})\leq U \leq\overline{Y}+t_{n-1,1-\alpha/2}*(S/\sqrt{n-1}) \qquad (7\text{-}1)$$

Some variables are defined below:

$\overline{Y}$ = the calculated mean value (that is, the sample mean) of the system performance criterion

S = the calculated standard deviation of the system performance criterion

n = the number of simulated values of the performance criterion used to calculate $\overline{Y}$ (and S)

U = the true mean value of the system performance criterion (which is unknown)

The confidence interval is 99% and in this example the number of samples is 50, so from Table 7-1, it can be obtained that $t_{49,0.995}=2.682$. By using eq. 7-1, it can be calculated that $13.06\leq U_{R.1}\leq 15.56$, $5.62\leq U_{R.2}\leq 7.74$, and $3.85\leq U_{R.3}\leq 5.31$. From the results of this calculation, we can predict with 99% confidence that the maximum current in any branch will not exceed the above upper limits; therefore the user may use these above statistical results to design and protect the circuit.

CHAPTER VIII

CONCLUSION

With the Gaussian distribution, the Monte-Carlo DC analysis offers an effective statistical analysis of a circuit giving maximums, minimums, means and standard deviations. All the statistical values of the input elements and output data such as node voltages, element voltages, element currents, and generalized outputs are available to the circuit designer. The MCDI post-processor redisplays the MCDC analysis results. System help files for the MCDC analysis and MCDI post-processor are provided in Appendix B.

Three major factors were considered in creating the programs - maximum execution speed, minimum memory, and maximum flexibility. The execution speed was increased by avoiding unnecessary or duplicated programming and by using IP DC to run the DC analysis. Avoiding putting the error messages in the Fortran subroutine and designing the error message in the MCDC and MCDC wexec macros saved some memory. A great flexibility was demonstrated by the MDRAND Fortran subroutine. Users can define arbitrary distributions to fit non-Gaussian distribution error tolerances of the elements. The example in Chapter VII of a power plant circuit using exponential distribution illustrates this flexibility.

Future development of Monte-Carlo capability for WATAND should include Monte-Carlo FR and TC analyses. The user can obtain the statistical performances of a circuit in more detail by employing the Monte-Carlo DC, FR, and TC analyses. Future work for Monte-Carlo analyses should be done to expand capability to more than 20 elements and parameters while balancing memory and speed requirements. Work might also be done to convert such analyses to built-in analyses by the WATAND development group at the University of Waterloo.

APPENDIX A

<u>COMPUTER RESULTS OF POWER DISTRIBUTION SYSTEM</u>

In this appendix, the entire computer results of a power plant circuit are listed.

```
mcdc wrtl r.1 1.1 r.2 1.2 r.3 1.3 *
dc ou i r.1 i r.2 i r.3 i sc.line ns 50

*** EXPONENTIAL DISTRIBUTION TO BE USED ***

** TIME **     * C #1 *     * C #2 *     * C #3 *

    1             82           46          153
 R VALUE      1.101D+01   1.000D+12   2.910D+01

    2             82          223          153
 R VALUE      1.000D+12   1.935D+01   2.849D+01

    3            594          223          153
 R VALUE      9.153D+00   2.034D+01   1.000D+12

    4            594          223          275
 R VALUE      9.623D+00   1.000D+12   2.618D+01

    5            594          322          275
 R VALUE      1.069D+01   2.182D+01   1.000D+12

    6            594          322         1068
 R VALUE      1.148D+01   1.000D+12   2.740D+01

    7            594          498         1068
 R VALUE      8.944D+00   1.000D+12   2.888D+01

    8            594          704         1068
 R VALUE      1.000D+12   2.137D+01   2.892D+01

    9            611          704         1068
 R VALUE      1.000D+12   2.036D+01   2.808D+01

   10            951          704         1068
 R VALUE      1.227D+01   1.000D+12   3.066D+01

   11            951          714         1068
 R VALUE      1.055D+01   1.000D+12   2.907D+01

   12            951         1213         1068
 R VALUE      1.000D+12   1.934D+01   3.000D+01

   13            963         1213         1068
 R VALUE      1.000D+12   1.985D+01   2.734D+01

   14            974         1213         1068
 R VALUE      1.000D+12   1.955D+01   2.929D+01

   15           1353         1213         1068
 R VALUE      9.902D+00   2.208D+01   1.000D+12

   16           1353         1213         1103
 R VALUE      1.318D+01   1.956D+01   1.000D+12

   17           1353         1213         1310
 R VALUE      1.029D+01   1.000D+12   2.940D+01

   18           1353         1524         1310
```

```
R VALUE      1.336D+01  1.935D+01  1.000D+12

   19          1353       1524       1698
R VALUE      1.000D+12  2.031D+01  2.903D+01

   20          1515       1524       1698
R VALUE      1.000D+12  2.066D+01  2.957D+01

   21          1822       1524       1698
R VALUE      9.633D+00  1.000D+12  3.044D+01

   22          1822       1808       1698
R VALUE      9.777D+00  2.174D+01  1.000D+12

   23          1822       1808       1733
R VALUE      8.285D+00  1.989D+01  1.000D+12

   24          1822       1808       1771
R VALUE      9.641D+00  2.112D+01  1.000D+12

   25          1822       1808       2748
R VALUE      8.309D+00  1.000D+12  2.826D+01

   26          1822       1899       2748
R VALUE      1.000D+12  2.005D+01  2.953D+01

   27          1854       1899       2748
R VALUE      1.000D+12  1.988D+01  3.193D+01

   28          2011       1899       2748
R VALUE      1.050D+01  1.000D+12  2.820D+01

   29          2011       2125       2748
R VALUE      1.000D+12  1.976D+01  3.071D+01

   30          2115       2125       2748
R VALUE      1.000D+12  2.000D+01  3.000D+01

   31          2700       2125       2748
R VALUE      1.166D+01  1.000D+12  2.972D+01

   32          2700       2662       2748
R VALUE      9.311D+00  1.000D+12  2.955D+01

   33          2700       2823       2748
R VALUE      1.000D+12  1.991D+01  2.893D+01

   34          2709       2823       2748
R VALUE      1.000D+12  1.958D+01  3.046D+01

   35          2867       2823       2748
R VALUE      7.839D+00  1.923D+01  1.000D+12

   36          2867       2823       2899
R VALUE      1.014D+01  1.000D+12  2.823D+01

   37          2867       3044       2899
R VALUE      1.000D+12  1.967D+01  3.024D+01

   38          2915       3044       2899
```

```
R VALUE        1.098D+01  1.977D+01  1.000D+12

   39             2915       3044       2992
R VALUE        1.000D+12  2.016D+01  3.180D+01

   40             2971       3044       2992
R VALUE        1.000D+12  2.084D+01  3.092D+01

   41             3253       3044       2992
R VALUE        1.079D+01  2.157D+01  1.000D+12

   42             3253       3044       3195
R VALUE        9.455D+00  1.000D+12  3.130D+01

   43             3253       3292       3195
R VALUE        1.040D+01  1.796D+01  1.000D+12

   44             3253       3292       3204
R VALUE        1.002D+01  2.073D+01  1.000D+12

   45             3253       3292       3267
R VALUE        1.000D+12  2.018D+01  3.102D+01

   46             3636       3292       3267
R VALUE        9.339D+00  2.200D+01  1.000D+12

   47             3636       3292       3340
R VALUE        1.049D+01  1.000D+12  2.948D+01

   48             3636       3335       3340
R VALUE        1.102D+01  1.000D+12  3.099D+01

   49             3636       3404       3340
R VALUE        8.172D+00  2.220D+01  1.000D+12

   50             3636       3404       3384
R VALUE        9.564D+00  2.140D+01  1.000D+12


OUTPUT             INITIAL       MINIMUM       MAXIMUM         MEAN          ST.DEV

I(R.1)          1.20000D+01  1.20000D-10  1.53090D+01  7.66201D+00  5.86808D+00
I(R.2)          6.00000D+00  1.20000D-10  6.68096D+00  4.02087D+00  2.76802D+00
I(R.3)          4.00000D+00  1.20000D-10  4.58392D+00  2.77109D+00  1.90680D+00
I(SC.LINE)      2.20000D+01  9.63805D+00  2.15499D+01  1.44540D+01  3.62550D+00

MCDC  EXECUTION TIME=    0.190 SEC.

# NOTICE # MCDC HAS SET DC ANALYSIS DEFAULTS TO 'KE NO IP ZERO'
```

APPENDIX B

<u>HELP FILES FOR MCDC AND MCDI</u>

In this appendix, the help files for the MCDC analysis and MCDI post-processor are offerred.

This user-defined Monte-Carlo analysis command finds minimum, maximum, mean, and standard-deviation values of DC-analyses outputs. Element and model-parameter values may be varied randomly.

```
+--------+-----------------------------------------------------------+
| MCDC   | WRTL  typ.nam1<.Vn> <m1 p1>   typ.nam2...                 |
|        |                                                           |
|        | WRTM  mtyp.mnam1  pnam1<.Vn> <m1 p1>  pnam2 ...           |
|        |         <END>  mtyp.mnam2 ...                             |
|        |                                                           |
|        | DC  OUtputs out-specs  <analy-specs>          <options>   |
+--------+-----------------------------------------------------------+
```

options: (defaults listed first)

```
        NS      50 | nsamp
        DSEED    0 | seed
```

where: (defaults)

WRTL specifies the linear elements to be randomly varied:

typ.nam	specifies linear element type and name;
Vn	is an optional value qualifier (n=1);
m,p	are numbers passed to the subroutine which generates random values for typ.nam. (m=0, p=0)

WRTM specifies which modeled element parameters are to be varied:

mtyp.mnam	specifies model element type and model (#Model) name;
pnam	specifies the parameter of the model to be varied;
Vn	is an optional parameter-value qualifier (n=1);
m,p	are numbers passed to the subroutine which generates values for pnam (m=0, p=0);
END	may be used to prevent ambiguity between a parameter name and an element type.

DC	Any normal DC-analysis parameters may be specified as desired, except that 'KEep NOne' is not allowed. Also, output action is automatically set to 'NONE' during MCDC execution.
NS	specifies the number of DC-analysis executions performed using random element/parameter values. (nsamp=50)
DSEED	specifies a seed value for random number generation. For seed =0, a random seed is generated from the CPU time. (seed=0)

Notes:

1. MCDC may be used only in the WEXEC environment. For long input lines it may be useful to write a WEXEC macro to call MCDC, and then make alterations in it as needed.

2. At least one WRTL or WRTM specification is required. At present, a maximum of 20 such specifications are possible.

3. MCDC analysis uses DC IP DC for each of its nsamp analyses. Only an initial DC analysis is affected by specifying DC IP ipname.

4. A DC analysis specification 'KEep what ON where' is used by MCDC to specify file name and mode of four files used to store input and output values. Using DC KE ALL requires MCDI to display specific output values.

5. Random values are generated by calls to subroutine MDRAND. Default MDRAND calls GGNQF, an IMSL Gaussian routine, but a replacement for MDRAND may be written and compiled. Parameters m and p may be used to send numerical values to the subroutine. For the default routine, m is the percent tolerance and specifies the 3-sigma point, while the initial valueserves as the mean; the default m=0 gives 10% tolerance.

Examples:

```
MCDC  NS 100  WRTM N.Q1 BF 20 IS 10  WRTL R.B R.E 15  *
     DC OUT I R.L V 20 40
MCDC  WRTL R.B 20 V.IN.V3 5  DC KE ALL IP DC OU V R.C
```

MCDI POST-PROCESSOR 7/29/87

This user-defined Monte-Carlo post-processor command reads kept results of an MCDC analysis and displays them according to parameters entered.

```
+--------+-----------------+------------------------------------------+
|  MCDI  |  INPUT  ON|OFF  |  ON/OFF value is saved after first use.  |
|        |  OU  out-specs  |  See below.                              |
|        |  <USE  fname>   |  File name of kept results.              |
|        |  <FM   fmode>   |  File mode of kept results.              |
+--------+-----------------+------------------------------------------+
```

where: (defaults)

INPUT	specifies whether the input data is to be displayed. INPUT is required the first time MCDI is run.
OU	specifies what outputs are to be displayed. OU is required to display MCDC results generated with DC KE ALL, but is ignored otherwise. Valid out-specs are node voltages and auxiliary variables. (See the display produced by DC PR OU ALL.) Up to seven outputs may be specified at one time.
USE	specifies the name of the kept files to be used. (fname=MCDC)
FM	specifies the file mode of the kept files. (fmode=A)

Note:

1. MCDI can be used in conjunction with MCDC to display more than seven output values. 'MCDC DC KE ALL' would be used once to generate the results, and then repeated use of MCDI will display up to seven outputs at one time. Only 'DC OU ALL' results are available with this approach.

Examples:

```
MCDI  INPUT OFF
MCDI  USE RUN1  OU V 1 V 25 V 17 38 I V.CC  INPUT ON
MCDI  OU V 10 V 20 V 30 40 I V.IN  USE RUN2  FM D
```

APPENDIX C

FORTRAN SUBROUTINES

In this appendix, the Fortran subroutines designed for the Monte-Carlo DC analysis and DI post-processor are listed in alphabetical order. Programming comments can be seen with each subroutine. The subroutine

MDCHCK is used in MCDC macro,

MDCHNG is called by subroutine MDMODL,

MDIALL is used in MCDI macro,

MDINPT is used MCDI macro,

MDINTL is used in MCDC macro,

MDIOUT is used in MCDC and MCDI macros,

MDMODL is called by MDREAD and MDRUSR,

MDPACK is called by subroutine MDREAD,

MDRAND is called by MDRUSR,

MDREAD is used in MCDC macro,

MDRUSR is called by #RU in MCDC macro,

MDSRCE is called by MDREAD and MDRUSR.

```
C*************************************************************************
      SUBROUTINE MDCHCK(IN,NIN,OUT,MOUT)
C*************************************************************************
C
C     MCDC ANALYSIS ROUTINE
C     CHECKS DC & DI ANALYSIS ERROR FLAG AND ENVIRONMENT OF WI OR WE
C
C     OUT(1) = 0 NO ERROR
C             -1 DC/DI NOT READY OR NOT IN WEXEC ENVIRONMENT
C     IN(1) = ANALYSIS TYPE NUMBER
C
C     CREATED 03-AUG-1987     J.F. SUEN  EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8 RDATA,IN(1),OUT(1)
C
      INTEGER IDATA(1),
     *        ANALYZ,
     *        WIWEFL,
     *        NIN,MOUT,ATP,APNTR
C
      EQUIVALENCE (RDATA(1),IDATA(1))
C
      COMMON /MEMORY/ RDATA(1)
     *       /APNTRS/ ANALYZ(30)
     *       /WIWEBL/ WIWEFL
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C...CHECK ANALYSIS ERROR FLAG
      IF(MOUT.LT.1) GOTO 998
      MOUT=1
      OUT(1)=0D0
      IF(IN(1).LE.0D0) GOTO 100
      ATP=IN(1)
      APNTR=ANALYZ(ATP)
      IF(IDATA(APNTR+1).NE.0) GOTO 999
      OUT(1)=-1D0
      GOTO 999
C
C...CHECK ENVIRONMENT
100   IF(WIWEFL.NE.2) OUT(1)=-1D0
      GOTO 999
998   MOUT=0
999   RETURN
      END
```

```
C**********************************************************************
      SUBROUTINE MDCHNG(PNAME,PLOC,PCOUNT,NFIXLP,INDEX,MIDA,INN,MPNAME,
     *                  MELN,MDATAS,TYPE,IER)
C**********************************************************************
C
C     MCDC ANALYSIS ROUTINE
C     ROUTINE TO DEAL WITH MODEL AND MODEL PARAMETER
C
C     TYPE = 1 FIND MODEL PARAMETER INITIAL VALUE
C            2 CHANGE MODEL PARAMETER VALUE TO RANDOM VALUE
C            3 CHECK IF MODEL PARAMETER EXIT AND FIND QUALIFIER VALUE
C            4 FROM INDEX VALUE TO FIND MODEL PARAMETER NAME
C     MPNAME = PARAMETER NAME FOUND
C     MDATAS = ELEMENT INITIAL VALUE/RANDOM VALUE
C     IER = 0 NO ERROR
C           1 QUALIFIER VALUE MISSING OR INVALID
C           2 MODEL PARAMETER CANNOT FIND
C
C     CREATED 03-AUG-1987     J.F. SUEN   EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8 RDATA,
     *       PNAME(1),MPNAME,MDATAS
C
      INTEGER IDATA(1),
     *        CONTSW,RECTER,TERLOC,
     *        CARD,CARD1,CSTART,CEND,CPNTR,
     *        PLOC(1),PCOUNT(1),MIDA(1),MNAME(13),
     *        PPTR,ETP,TYPE,BLANK,INN,INDEX,NFIXLP,I1,I2
C
      COMMON /MEMORY/ RDATA(1)
     *       /RDQCBL/ CONTSW,RECTER,TERLOC
     *       /BCARD/  CARD(80),CARD1(80),CSTART,CEND,CPNTR
C
      DATA BLANK/' '/
C
      EQUIVALENCE (RDATA(1),IDATA(1))
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      ETP=MIDA(1)
      PPTR=MIDA(5)-1
      I=PPTR+INDEX
      GOTO (5,10,20,50),TYPE
C
C...TYPE EQUAL 1
5     MDATAS=RDATA(I)
      GOTO 996
C
C...TYPE EQUAL 2
10    RDATA(I)=MDATAS
      DO 15 J=1,80
      CARD(J)=BLANK
15    CONTINUE
      CPNTR=CSTART
      RECTER=2
      CALL AMSECT(ETP,MELN)
      GOTO 996
C
C...TYPE EQUAL 3
20    DO 30 J=1,NFIXLP
```

```
      IF(MPNAME.EQ.PNAME(J))GO TO 40
30    CONTINUE
      GOTO 997
40    IF(INN.GT.PCOUNT(J))GO TO 998
      INDEX=PLOC(J)+INN-1
      GOTO 996
C
C...TYPE EQUAL 4
50    DO 60 J=1,NFIXLP
      IF(INDEX.LT.PLOC(J)) GOTO 70
      IF(INDEX.EQ.PLOC(J)) GOTO 80
60    CONTINUE
      GOTO 80
70    J=J-1
80    INN=INDEX-PLOC(J)+1
      MPNAME=PNAME(J)
C
C...TERMINATION
996   IER=0
      GOTO 999
997   IER=2
      GOTO 999
998   IER=1
999   RETURN
      END
```

```
C**********************************************************************
      SUBROUTINE MDIALL(IN,NIN,OU,MOUT)
C**********************************************************************
C
C     ROUTINE TO DISPLAY THE 'KEEP ALL' DISK FILE DATA
C
C     IN(1) = #OF SAMPLE IN MCDC ANALYSIS
C     IN(2) = OUCODE = 1 KE OU
C                    = 2 KE ALL
C     IN(4) = TIME OF START TIMER
C     IN(5) = CODE = 1 INPUT DISPLAY ON
C                  = 0 INPUT DISPLAY OFF
C     OUT(1) = -1 ERROR
C            >  0 DISK READING TIME DIFFERENT WITH #OF SAMPLE
C            =  0 CORRECT
C     IN(3) NOT USED
C
C     CREATED 03-AUG-1987      J.F. SUEN   EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*4 TIME
      REAL*8 RDATA,
     *       OUTPV,
     *       DATE,
     *       FNAME,
     *       LVLID,PRIVID,
     *       WDJTID,
     *       IN(1),OU(1),TSTART,TEND,CLOCK,BLANK8,SUMW(8),
     *       ORIG(8),AVG(8),SDEV(8),LARG(8),SMAL(8),SUM(8)
C
      INTEGER IDATA(1),
     *        RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW,
     *        ANALYZ,
     *        EDITID,
     *        HEAD,ETOUSP,NOUT,ALLCD,XLBLCD,CMPLCD,XANDCD,
     *        OUTPTP,
     *        ATTCD,
     *        WDJTFL,
     *        OUPNTR,OUPNTT,
     *        ATP,APNTR,IPTR,RPTR,IER,XLOGCD,COMPSW,EXSAVD(5),
     *        NS,FLAG,OUCODE,NIN,MOUT,CODE,ITOTAL
C
      LOGICAL*1 EXBUFR(80)
C
      EQUIVALENCE (RDATA(1),IDATA(1))
C
      COMMON /MEMORY/ RDATA(1)
     *       /BINPUT/ RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW
     *       /APNTRS/ ANALYZ(30)
     *       /BLCK07/ LVLID,PRIVID,EDITID
     *       /OUTBL1/ HEAD(12,7),ETOUSP(3,8),NOUT,ALLCD,XLBLCD,CMPLCD,
     *                XANDCD
     *       /OUTBL2/ OUTPV(8),OUTPTP(8)
     *       /DATEBL/ DATE(3)
     *       /FNBLOK/ FNAME
     *       /ATTIBL/ ATTCD
     *       /WDJTBL/ WDJTID,WDJTFL
     *       /OUORBL/ OUPNTR(8),OUPNTT(8)
C
      DATA BLANK8/'  '/
```

```
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      FLAG=0
      IER1=0
      NS=0
      MOUT=1
      OU(1)=0
      ITOTAL=IN(1)
      OUCODE=IN(2)
      TSTART=IN(4)
      CODE=IN(5)
      IF(OUCODE.EQ.1) GOTO 999
      DO 10 I=1,8
      SUM(I)=0.0
10    SUMW(I)=0.0
      ATP=12
      APNTR=ANALYZ(ATP)
      IPNTR=APNTR+10
      RPTR=IDATA(APNTR)
      NOUT=IDATA(APNTR+7)
C
C...INITIALIZE READING
      CALL KUINIT(APNTR+10,IDATA(APNTR+7),APNTR+31,RDATA(RPTR),EXSAVD,
     *            EXBUFR,IER)
      IF(IER.NE.0) GOTO 900
      CALL PACKCH(EXBUFR(6),DATE,20,1,1)
C
C...CHECK OUTPUT PARAMETER
20    I=1
60    IF(IDATA(OUPNTR(I)).LT.0) GOTO 991
      PNTR=OUPNTR(I)
      PNTR=PNTR+1
      IF(IDATA(PNTR).NE.1) GOTO 991
      PNTR=PNTR+2
      IT=IDATA(PNTR)
      IF(IT.NE.0) GOTO 991
      PNTR=PNTR+1
      IT=IDATA(PNTR)
      IF(IT.NE.0) GOTO 991
      I=I+1
      IF(I.LE.NOUT) GOTO 60
70    IF(ATTCD.EQ.1) GOTO 150
      CALL KURECE(IER)
      IF(IER.NE.0) GOTO 100
      IF(FLAG.EQ.1) GOTO 90
      DO 80 I=1,NOUT
      LARG(I)=OUTPV(I)
80    SMAL(I)=OUTPV(I)
90    FLAG=1
      GOTO 70
C
C...ANY MORE SETS?
100   CALL KUNEXT(IER)
      IF(IER.NE.0) GOTO 150
      NS=NS+1
      DO 120 I=1,NOUT
      SUM(I)=SUM(I)+OUTPV(I)
      SUMW(I)=SUMW(I)+OUTPV(I)**2
      IF(OUTPV(I).GT.LARG(I)) LARG(I)=OUTPV(I)
      IF(OUTPV(I).LT.SMAL(I)) SMAL(I)=OUTPV(I)
```

```
120   CONTINUE
      GOTO 70
C
C...TERMINATION
150   IF(NS.NE.ITOTAL) IER1=1
      DO 230 I=1,NOUT
      AVG(I)=SUM(I)/NS
      SDEV(I)=(SUMW(I)/NS-AVG(I)**2)**(1.0/2.0)
230   ORIG(I)=OUTPV(I)
      IF(CODE.EQ.1) GOTO 240
      WRITE(WRUNT,1400)LVLID,EDITID,PRIVID,DATE,WDJTID,FNAME
240   WRITE(WRUNT,1100)
      DO  250  J=1,NOUT
      CALL OUTHDG(HEAD(1,J),IDATA(IPNTR))
      WRITE(WRUNT,1200)(HEAD(I,J),I=1,12),ORIG(J),SMAL(J),LARG(J),AVG(J)
     *                 ,SDEV(J)
250   IPNTR=IPNTR+3
      CALL GTIMER(TIME)
      TEND=TIME
      CLOCK=TEND-TSTART
      WRITE(WRUNT,1300)CLOCK
      IF(IER1.NE.0) GOTO 992
      GOTO 900
991   OU(1)=-1
      GOTO 900
992   OU(1)=NS
900   CALL KUTERM
      RDATA(RPTR)=BLANK8
999   RETURN
1100  FORMAT(//' OUTPUT              INITIAL       MINIMUM       MAXIMUM
     *   MEAN          ST.DEV'/)
1200  FORMAT(1X,12A1,2X,1P5E13.5)
1300  FORMAT(/' DISPLAY TIME=',F8.3,' SEC.'/)
1400  FORMAT(/1X,'MCDC',1X,A6,A2,1X,A8,1X,2A8,A4,A8,'  FILE: ',A8)
      END
```

```
C***********************************************************************
      SUBROUTINE MDINPT(IN,NIN,RSUL,MOUT)
C***********************************************************************
C
C     MCDI POST-PROCESSOR ROUTINE
C     DISPLAYS INPUT INFORMATION FOR MCDC ANALYSIS
C
C     IN(1) = IDCODE = 1 INPUT DISPLAY ON
C                    = 0 INPUT DISPLAY OFF
C     IN(2) = PCODE  = 1 OUTPUT SPECIUFIED
C                    = 0 NO OUTPUT SPECIFIED
C     RSUL(1) = #OF SAMPLE IN MCDC ANALYSIS
C             = -1 NOT IN THE WE ENVIRONMENT
C     RSUL(2) = OUCODE = 1 KE OU IN MCDC ANALYSIS
C                      = 2 KE ALL IN MCDC ANALYSIS
C     RSUL(4) = TIME OF TIMER START
C     RSUL(3) NOT USED
C
C     CREATED 03-AUG-1987     J.F. SUEN  EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*4  TIME
      REAL*8  RDATA,
     *        LVLID,PRIVID,
     *        WDJTID,
     *        FNAME,
     *        FILEID(3),FILEIP(3),
     *        BUFFER(10),IN(1),RSUL(1)
C
      INTEGER IDATA(1),
     *        EDITID,
     *        WDJTFL,
     *        RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW,
     *        ANALYZ,
     *        NIN,MOUT,RECON,IDCODE,OUCODE,PCODE,
     *        LS,LT,NS,ITOTAL,APNTR,RPTR
C
      EQUIVALENCE  (RDATA(1),IDATA(1))
C
      COMMON /MEMORY/ RDATA(1)
     *       /BLCK07/ LVLID,PRIVID,EDITID
     *       /WDJTBL/ WDJTID,WDJTFL
     *       /FNBLOK/ FNAME
     *       /BINPUT/ RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW
     *       /APNTRS/ ANALYZ(30)
C
      DATA FILEID/'MCDC','MDKPINFO','A'/,
     *     FILEIP/'MCDC','MDKPVALS','A'/
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IF(MOUT.LT.4) GOTO 997
      IDCODE=IN(1)
      PCODE=IN(2)
      CALL GTIMER(TIME)
      RSUL(3)=0.0
      RSUL(4)=TIME
C
C...FIND FILE NAME AND FILE TYPE
      APNTR=ANALYZ(12)
      RPTR=IDATA(APNTR)
      FILEID(1)=RDATA(RPTR)
      FILEIP(1)=RDATA(RPTR)
      FILEID(3)=RDATA(RPTR+1)
      FILEIP(3)=RDATA(RPTR+1)
C
C...READ TITLE INFORMATION
      CALL IOOPEN(FILEID,1,10*8,RECON,IER)
      IF(IER.NE.0) GOTO 995
      CALL IOREAD(FILEID,RECON,BUFFER,10*8,IER)
      IF(IER.EQ.2) GOTO 995
      CALL IOCLOS(FILEID,1,IER)
      IF(IER.NE.0) GOTO 995
      NS=BUFFER(2)
      OUCODE=BUFFER(3)
      IF(PCODE.EQ.0.AND.OUCODE.EQ.2D0) GOTO 990
      ITOTAL=BUFFER(1)
      IF(IDCODE.EQ.0.AND.OUCODE.EQ.2D0) GOTO 990
      WRITE(WRUNT,1400)LVLID,EDITID,PRIVID,(BUFFER(J),J=8,10),
     *                 WDJTID,FNAME
      IF(IDCODE.EQ.0) GOTO 990
      WRITE(WRUNT,1300)NS,(BUFFER(J),J=4,7)
C
C...READ DISK INPUT INFORMATION
      CALL IOOPEN(FILEIP,1,8*8,RECON,IER)
      IF(IER.NE.0) GOTO 995
      WRITE(WRUNT,1100)
      DO 70 I=1,ITOTAL
      CALL IOREAD(FILEIP,RECON,BUFFER,8*8,IER)
      IF(IER.EQ.2) GOTO 995
70    WRITE(WRUNT,1200)(BUFFER(J),J=1,3),(BUFFER(J),J=4,8)
      CALL IOCLOS(FILEIP,1,IER)
      IF(IER.NE.0) GOTO 995
990   RSUL(1)=NS
      RSUL(2)=OUCODE
      MOUT=4
      GOTO 999
995   RSUL(1)=-1
      MOUT=1
      GOTO 999
997   MOUT=0
999   RETURN
1100  FORMAT(' INPUT                          INITIAL    MINIMUM    MAXIMUM
     *    MEAN      ST.DEV'/)
1200  FORMAT(1X,3A8,1P5E11.3)
1300  FORMAT(/5X,'NUMBER OF SAMPLES = ',I4,9X,4A8/)
1400  FORMAT(/1X,'MCDC',1X,A6,A2,1X,A8,1X,2A8,A4,A8,'  FILE: ',A8)
      END
```

```
C***********************************************************************
      SUBROUTINE MDINTL(IN,NIN,RSLT,MOUT)
C***********************************************************************
C
C     MCDC ANALYSIS ROUTINE
C     CHECKS THE 'WRTL' 'WRTM' PART AND DC ANALYSIS SPECIFIES
C
C     RSLT(1) = OUCODE = 1 FOR 'KE OU'
C                      = 2 FOR 'KE ALL'
C                      =-1 ERROR FOR 'OU ALL'
C                      =-2 ERROR FOR NOT USE 'KE OU/KE ALL'
C     RSLT(2) = #OF AFTER WRTL
C     RSLT(3) = #OF AFTER WRTM
C     RSLT(4) = TIME OF TIME START
C
C     CREATED 03-AUG-1987     J.F. SUEN  EE DEP. YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*4 TIME
      REAL*8 RDATA,
     *       VALNAM,
     *       CNDC,IN(1),RSLT(1),CNWRT(2)
C
      INTEGER IDATA(1),
     *        LENHED,LINK,FREEH,
     *        MACROH,GLBLPH,GLBLNH,MGLPHP,LCLPH,
     *        ANALYZ,
     *        APNTR,KEPTR,
     *        IWRT(2),OUCODE,
     *        IT,IV,NIN,MOUT,IER
C
      EQUIVALENCE (RDATA(1),IDATA(1))
C
      COMMON /MEMORY/ RDATA(1)
     *       /MCLIST/ VALNAM(750),LENHED(750),LINK(750),FREEH
     *       /MCHEDS/ MACROH,GLBLPH,GLBLNH,MGLPHP,LCLPH
     *       /APNTRS/ ANALYZ(30)
C
      DATA CNWRT/'WRTL','WRTM'/,CNDC/'DC'/
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      CALL GTIMER(TIME)
      OUCODE=1
      MOUT=4
      RSLT(4)=TIME
C
C...CHECK 'WRTL' AND 'WRTM' PART
      IWRT(1)=0
      IWRT(2)=0
      DO 60 I=1,2
      IT=LCLPH
10    IF(VALNAM(IT).EQ.CNWRT(I)) GOTO 20
      IT=LINK(IT)
      GOTO 10
20    IF(LENHED(IT).EQ.0) GOTO 50
      IV=LENHED(IT)
30    IF(LENHED(IV).EQ.0) GOTO 40
      IWRT(I)=IWRT(I)+1
40    IF(LINK(IV).EQ.0) GOTO 50
      IV=LINK(IV)
      GOTO 30
50    RSLT(I+1)=IWRT(I)
60    CONTINUE
C
C...CHECK FOR 'DC OU  ALL' AND KE SPECS
      APNTR=ANALYZ(1)
      KEPTR=IDATA(APNTR+35)
C
C...CHECK FOR DC 'OU ALL'
      IF(IDATA(APNTR+9).EQ.1) GOTO 997
C
C...CHECK FOR 'KE NONE'
      IF(KEPTR.EQ.0) GOTO 998
C
C...CHECK FOR ANY SPECIFIED OUTPUTS
      IF(IDATA(KEPTR+3).NE.0) GOTO 998
C
C...CHECK FOR 'KE ALL OU'
      IF(IDATA(KEPTR+1).NE.0.AND.IDATA(KEPTR+2).NE.0) GOTO 998
C
C...CHECK FOR 'KE ALL'
      IF(IDATA(KEPTR+1).NE.0) OUCODE=2
      RSLT(1)=OUCODE
      GOTO 999
C
997   RSLT(1)=-1
      GOTO 999
998   RSLT(1)=-2
999   RETURN
      END
```

```
C***********************************************************************
      SUBROUTINE  MDIOUT(IN,NIN,OUT,MOUT)
C***********************************************************************
C
C     MCDC AND MCDI ANALYSIS ROUTINE
C     CALCULATES AND DISPLAYS STASTICAL RESULTS OF OUTPUTS
C
C     IN(1) = TCODE = 1 CALLED FROM MCDC
C                     2 CALLED FROM MCDI
C     IN(2) = NS = #OF SAMPLES
C     IN(3) = OUCODE = 1 FOR KE OU
C                    = 2 FOR KE ALL - NO OUTPUT HERE
C     IN(6) = TSART = START CPU TIME
C     IN(4),IN(5) NO USE
C     OUT(1) = 0 NO ERROR
C            =-1 ERROR OCCUR
C
C     CREATED  03-AUG-1987     J.F. SUEN   EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*4 TIME
      REAL*8 RDATA,
     *       FILEID(3),FILEIR(3),
     *       IN(1),OUTPUT(8),ERR1(2),
     *       OUT(1),SDEV(8),SUMW(8),SUM(8),AVG(8),SMAL(8),LARG(8),
     *       ORIGIN(8),BUFFER(5),DSQRT,DABS,
     *       TSTART,TEND,CLOCK,BLANK8
C
      INTEGER IDATA(1),
     *        RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW,
     *        HEAD,ETOUSP,NOUT,ALLCD,XLBLCD,CMPLCD,XANDCD,
     *        ANALYZ,
     *        LNAME(13),RECNO,RECON,INUM,APNTR,KEPTR,ATP,
     *        NS,MOUT,NIN,IER,OUCODE,TCODE,BLANK,RPTR
C
      EQUIVALENCE (RDATA(1),IDATA(1)),(BLANK8,BLANK)
C
      COMMON /MEMORY/ RDATA(1)
     *       /BINPUT/ RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW
     *       /OUTBL1/ HEAD(12,7),ETOUSP(3,8),NOUT,ALLCD,XLBLCD,CMPLCD,
     *                XANDCD
     *       /APNTRS/ ANALYZ(30)
C
      DATA FILEID/'MCDC','WKEEP','A'/,
     *     FILEIR/'MCDC','WKEEPID','A'/,BLANK8/' '/
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C...CHECK SPECIFIED DISK BEEN ASSIGNED
      RECNO=0
      RECON=0
      MOUT=1
      ATP=1
      TCODE=IN(1)
      IF(TCODE.EQ.2) ATP=12
      APNTR=ANALYZ(ATP)
      NS=IN(2)
      OUCODE=IN(3)
      TSTART=IN(6)
      OUT(1)=0D0
      IF(OUCODE.EQ.2) GOTO 240
      IF(TCODE.EQ.2) GOTO 300
```

```
C...FIND FILE NAME AND FILE TYPE
      KEPTR=IDATA(APNTR+35)
      FILEID(1)=RDATA(IDATA(KEPTR))
      FILEIR(1)=RDATA(IDATA(KEPTR))
      FILEID(3)=RDATA(IDATA(KEPTR)+1)
      FILEIR(3)=RDATA(IDATA(KEPTR)+1)
      N=NS+1
      K=NOUT+1
      DO 10 I=1,K
      OUTPUT(I)=0.0
      AVG(I)=0.0
      SUM(I)=0.0
      SUMW(I)=0.0
10    SDEV(I)=0.0
C
C...OPEN IO AND READ DATA
110   CALL IOOPEN(FILEID,1,K*8,RECNO,IER)
      IF(IER.NE.0) GOTO 998
      DO 150 J=1,N
      CALL IOREAD(FILEID,RECNO,OUTPUT,K*8,IER)
      IF(IER.EQ.2) GOTO 998
      IF(J.EQ.N) GOTO 170
      DO 150 I= 1,K
      SUM(I)=SUM(I)+OUTPUT(I)
      SUMW(I)=SUMW(I)+OUTPUT(I)**2
      IF(J.NE.1) GOTO 120
      LARG(I)=OUTPUT(I)
      SMAL(I)=OUTPUT(I)
      GOTO 150
120   IF(OUTPUT(I).GT.LARG(I)) LARG(I)=OUTPUT(I)
      IF(OUTPUT(I).LT.SMAL(I)) SMAL(I)=OUTPUT(I)
150   CONTINUE
170   DO 180 I=1,K
      AVG(I)=SUM(I)/NS
      SDEV(I)=SUMW(I)/NS-AVG(I)**2
      SDEV(I)=DABS(SDEV(I))
      SDEV(I)=DSQRT(SDEV(I))
180   ORIGIN(I)=OUTPUT(I)
      CALL IOCLOS(FILEID,1,IER)
      IF(IER.NE.0) GOTO 998
C
C...PREPARE THE OUTPUT
      CALL IOOPEN(FILEIR,0,13*4,1,IER)
      IF(IER.NE.0) GOTO 998
      CALL IOOPEN(FILEID,0,5*8,1,IER)
      IF(IER.NE.0) GOTO 998
      WRITE(WRUNT,1100)
      DO 230 J=1,NOUT
      L=J+1
      DO 210 I=1,12
      LNAME(I)=BLANK
210   LNAME(I)=HEAD(I,J)
      LNAME(13)=NOUT
      BUFFER(1)=ORIGIN(L)
      BUFFER(2)=SMAL(L)
      BUFFER(3)=LARG(L)
      BUFFER(4)=AVG(L)
      BUFFER(5)=SDEV(L)
      WRITE(WRUNT,1200)(HEAD(I,J),I=1,12),ORIGIN(L),
     *      SMAL(L),LARG(L),AVG(L),SDEV(L)
```

```
      CALL IOWRIT(FILEIR,LNAME,13*4,IER)
      IF(IER.NE.0) GOTO 998
      CALL IOWRIT(FILEID,BUFFER,5*8,IER)
      IF(IER.NE.0) GOTO 998
230   CONTINUE
      CALL IOCLOS(FILEIR,0,IER)
      IF(IER.NE.0) GOTO 998
      CALL IOCLOS(FILEID,0,IER)
      IF(IER.NE.0) GOTO 998
240   CALL GTIMER(TIME)
      TEND=TIME
      CLOCK=TEND-TSTART
      WRITE(WRUNT,1300)CLOCK
      GOTO 999
C
C...BEGIN MCDI SECTION
C...READ DATA FROM DISK
300   RPTR=IDATA(APNTR)
      FILEID(1)=RDATA(RPTR)
      FILEIR(1)=RDATA(RPTR)
      FILEID(3)=RDATA(RPTR+1)
      FILEIR(3)=RDATA(RPTR+1)
      RDATA(RPTR)=BLANK8
      CALL IOOPEN(FILEIR,1,13*4,RECNO,IER)
      IF(IER.NE.0) GOTO 998
      CALL IOOPEN(FILEID,1,5*8,RECON,IER)
      IF(IER.NE.0) GOTO 998
      CALL IOREAD(FILEIR,RECNO,LNAME,13*4,IER)
      IF(IER.EQ.2) GOTO 998
      CALL IOREAD(FILEID,RECON,BUFFER,5*8,IER)
      IF(IER.NE.0) GOTO 998
      INUM=LNAME(13)
      WRITE(WRUNT,1100)
      DO 370 I=1,INUM
      WRITE(WRUNT,1200)(LNAME(J),J=1,12),(BUFFER(J),J=1,5)
      IF(I.EQ.INUM) GOTO 370
      DO 360 J=1,12
360   LNAME(J)=BLANK
      CALL IOREAD(FILEIR,RECNO,LNAME,13*4,IER)
      IF(IER.EQ.2) GOTO 998
      CALL IOREAD(FILEID,RECON,BUFFER,5*8,IER)
      IF(IER.EQ.2) GOTO 998
370   CONTINUE
      CALL IOCLOS(FILEID,1,IER)
      IF(IER.NE.0) GOTO 998
      CALL IOCLOS(FILEIR,1,IER)
      IF(IER.NE.0) GOTO 998
      CALL GTIMER(TIME)
      TEND=TIME
      CLOCK=TEND-TSTART
      WRITE(WRUNT,1400)CLOCK
      GOTO 999
998   OUT(1)=-1
999   RETURN
1100  FORMAT(//' OUTPUT            INITIAL      MINIMUM      MAXIMUM
     *   MEAN        ST.DEV   '/)
1200  FORMAT(1X,12A1,2X,1P5E13.5)
1300  FORMAT(/' MCDC  EXECUTION TIME=',F8.3,' SEC.'/)
1400  FORMAT(/' DISPLAY TIME=',F8.3,' SEC.'/)
      END
```

```
C***********************************************************************
      SUBROUTINE MDMODL(VCHANG,PNAME,ELN,ETP,INDEX,INN,TYPE,IER)
C***********************************************************************
C
C     MCDC ANALYSIS ROUTINE
C     ROUTINE CONTROLS CALLS TO MDCHNG
C
C     TYPE = 1 FIND INITIAL VALUE OF MODEL PARAMETER
C          = 2 CHANGE MODEL PARAMETER BY RANDOM NUMBER
C          = 3 FIND QUALIFIER VALUE
C          = 4 FIND PARAMETER NAME AND QUALIFIER FOR DISPLAY
C
C     CREATED 03-AUG-1987     J.F. SUEN   EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8 RDATA,PNAME,VCHANG,
     *       EPAME(11),DPAME(7),GPAME(11)
C
      INTEGER IDATA(1),IREP,UIPTR,TYPE,
     *        BTYPE,NTYPE,MTYPE,MIDAPT,ELN,ETP,INDEX,
     *        EDIDA,MDIPTR,MDRPTR,NDIM,NPAR,NFIXLP,
     *        EPLOC(11),EPCUNT(11),
     *        DPLOC(7),DPCUNT(7),
     *        GPLOC(11),GPCUNT(11)
C
      EQUIVALENCE (RDATA(1),IDATA(1))
C
      COMMON /MEMORY/ RDATA(1)
     *       /UDELEM/ UIPTR(20)
     *       /BLCK00/ BTYPE,NTYPE,MTYPE
C
      DATA EPAME/'IS','IF','BR','CS','GO','TE','TS','BF','FT','CE',
     *       'CC'/ ,
     *     EPLOC/  1 ,  2 ,  3 ,  4 ,  5 ,  6 ,  7 ,  8 , 14 , 18 ,
     *       22 / ,
     *     EPCUNT/ 1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  6 ,  4 ,  4 ,
     *        4 / ,
     *     DPAME/'I1','I2','TE','RS','VB','CJ','TA'/ ,
     *     DPLOC/  1 ,  3 ,  5 ,  6 ,  7 ,  8 , 12 / ,
     *     DPCUNT/ 2 ,  2 ,  1 ,  1 ,  1 ,  4 ,  1 / ,
     *     GPAME/'IS','IE','IC','TE','BF','BR','QB','CE','TF','CC',
     *       'TR'/ ,
     *     GPLOC/  1 ,  2 ,  4 ,  6 ,  7 ,  8 ,  9 , 13 , 17 , 20 ,
     *       24 / ,
     *     GPCUNT/ 1 ,  2 ,  2 ,  1 ,  1 ,  1 ,  4 ,  4 ,  3 ,  4 ,
     *        3 /
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C...MODEL PARAMETER SEARCHING OR CHANGING VALUE
      MIDAPT=ELN-3
      IF(ETP.NE.11.AND.ETP.NE.12) GOTO 144
      CALL MDCHNG(EPAME,EPLOC,EPCUNT,11,INDEX,IDATA(MIDAPT),INN,PNAME,
     *         ELN,VCHANG,TYPE,IER)
      GOTO 999
 144  IF(ETP.NE.13) GOTO 146
      CALL MDCHNG(DPAME,DPLOC,DPCUNT,7,INDEX,IDATA(MIDAPT),INN,PNAME,
     *         ELN,VCHANG,TYPE,IER)
      GOTO 999
 146  IF(ETP.NE.14.AND.ETP.NE.15)GOTO 148
      CALL MDCHNG(GPAME,GPLOC,GPCUNT,11,INDEX,IDATA(MIDAPT),INN,PNAME,
     *         ELN,VCHANG,TYPE,IER)
```

```
      GOTO 999
 148  EDIDA=UIPTR(ETP-BTYPE)
      MDIPTR=IDATA(EDIDA+16)+EDIDA
      MDRPTR=IDATA(MDIPTR)
      NPAR=IDATA(MDIPTR+1)
      NDIM=IDATA(EDIDA+4)
      NFIXLP=NPAR-NDIM
      CALL MDCHNG(RDATA(MDRPTR+2),IDATA(MDIPTR+5+NPAR),IDATA(MDIPTR+5),
     *     NFIXLP,INDEX,IDATA(MIDAPT),INN,PNAME,ELN,VCHANG,TYPE,IER)
999   RETURN
      END
```

```
C***********************************************************************
      SUBROUTINE MDPACK(IPAR,ETP,ELN,INDEX,NAME,POSPTR,MFLAG,TYPE,IER)
C***********************************************************************
C
C     MCDC ANALYSIS ROUTINE CALLED BY MDREAD
C     FINDS ELEMENT TYPE, POSITION LINE, AND QUALIFIER VALUE
C
C     TYPE = 1  FIND ELEMENT TYPE ETP
C          = 2  FIND ELEMENT LINE ELN OR PARAMETER NAME
C          = 3  FIND QUALIFIER VALUE (INDEX)
C     IER = 0 NO ERROR
C         = 1 OR 2 ERROR
C
C     CREATED  03-AUG-1987     J.F. SUEN  EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8  BLANK8,NAME
C
      INTEGER IPAR(24),DIGIT(10),VNAME,LETV,BLANK,DOT,TYPE,
     *        ETP,ELN,POSPTR
C
      DATA BLANK8/' '/,LETV/'V'/,DOT/'.'/,
     *     DIGIT /'0','1','2','3','4','5','6','7','8','9'/
C
      EQUIVALENCE (BLANK8,BLANK)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IF(TYPE-2) 10,50,150
C
C...FIND ELEMENT/MODEL TYPE
10    N=0
      I=0
20    I=I+1
      IF(N.GT.4) GOTO 997
      IF(IPAR(I).EQ.DOT) GOTO 30
      N=N+1
      GOTO 20
30    VNAME=BLANK
      CALL PACKCH(IPAR(1),VNAME,N,4,1)
      CALL CHETYP(VNAME,ETP)
      POSPTR=I
      GOTO 998
C
C...FIND ELEMENT/MODEL POSITION LINE
50    I=POSPTR
      N=0
60    I=I+1
      IF(N.GT.8) GOTO 997
      IF(IPAR(I).EQ.DOT.OR.IPAR(I).EQ.BLANK) GOTO 70
      N=N+1
      GOTO 60
70    NAME=BLANK8
      IF(MFLAG-2) 80,90,100
80    CALL PACKCH(IPAR(I-N),NAME,N,4,1)
      CALL FINDE(ETP,NAME,ELN)
      GOTO 110
90    CALL PACKCH(IPAR(I-N),NAME,N,4,1)
      CALL FINDM(ETP,NAME,ELN)
      GOTO 110
100   CALL PACKCH(IPAR(1),NAME,N,4,1)
```

```
      CALL FINDM(ETP,NAME,ELN)
110   POSPTR=I
      GOTO 998
C
C...FIND QUALIFIER VALUE
150   IF(IPAR(POSPTR).EQ.BLANK) GOTO 998
      I=POSPTR+1
      IF(IPAR(I).NE.LETV) GOTO 997
      INDEX=0
      N=0
160   I=I+1
      N=N+1
      IF(N.GT.6) GOTO 997
      IF(IPAR(I).EQ.BLANK.AND.N.NE.1) GOTO 998
      DO 170 J=1,10
      IF(IPAR(I).EQ.DIGIT(J)) GOTO 180
170   CONTINUE
      GOTO 997
180   INDEX=10*INDEX+J-1
      GOTO 160
997   IER=1
      GOTO 999
998   IER=0
999   RETURN
      END
```

```
C*************************************************************************
      SUBROUTINE MDRAND(RANVAL,TITLE,INPFL,IREP,ICOUNT,DSEED,INIVL,
     *                  PAR1,PAR2,IER)
C*************************************************************************
C
C     MCDC ANALYSIS ROUTINE
C     ROUTINE TO CREATE THE RANDOM NUMBER FOR GAUSSIAN DISTRIBUTION
C
C     IREP    CURRENT TIME OF MDRUKE SUBROUTINE BEEN CALLED
C     DSEED   SEED NUMBER OF RANDOM NUMBER GENERATOR
C     RANVAL  RANDOM NUMBER BRING BACK TO MDRUKE SUBROUTINE
C     INIVL   THE INITIAL VALUE OF THE CHANGED ELEMENT
C     PAR1,PAR2  NUMBERS FOLLOWING ELEMENT
C                IF PAR1=0  SET DEFAULT ERROR TOLERANCE 10%
C     ICOUNT  # OF LINEAR AND NON-LINEAR ELEMENTS
C     TITLE   USER CAN ASSIGN IT (32 BYTES LONG)
C     INPFL  =1 DISPLAY THE INPUT INFORMATION
C           ^=1 NOT DISPLAY THE INPUT INFORMATION
C     IER    = 0 NO ERROR
C           ^= 0 ERROR OCCUR ANALYSIS STOP
C
C     CREATED  03-AUG-1987  14:15:36     J.F. SUEN   EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8  DSEED,RANVAL,INIVL,PAR1,PAR2,SDEV,DABS,TPAR1,
     *        TITLE(4),TITAL(4)
C
      REAL    R,GGNQF
C
      INTEGER IREP,ICOUNT,INPFL,IER
C
      DATA TITAL /'     GAU','SSIAN  D','ISTRIBUT','ION     '/
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IF(IREP.GT.1) GOTO 20
      DO 10 J=1,4
10    TITLE(J)=TITAL(J)
20    TPAR1=PAR1
      IF(PAR1.EQ.0D0) TPAR1=10D0
      SDEV=(INIVL*DABS(TPAR1)/100D0)*2D0/6D0
      R=GGNQF(DSEED)
      RANVAL=INIVL+R*SDEV
      IER=0
      RETURN
      END
```

```
C***********************************************************************
      SUBROUTINE MDREAD(IN,NIN,ELMOUT,MOUT)
C***********************************************************************
C
C     MCDC ANALYSIS ROUTINE
C     DECODES INPUT WRTL AND WRTM VARIABLES
C
C     ELMOUT(1) = IWRTL = # OF LINEAR ELEMENT
C               = -1 ERROR
C     ELMOUT(2) = IWRTM = # OF NON-LINEAR ELEMENT
C     IN(1) = OUCODE
C     IN(2) = #OF VARIABLES IN WRTL
C     IN(3) = #OF VARIABLES IN WRTM
C     MOUT = 5 * (IWRTL+IWRTM) +2
C     ELMOUT = ELEMENT/PARAMETER INFO
C
C     CREATED  03-AUG-1987     J.F. SUEN  EE DEP. YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8 RDATA,
     *       VALNAM,
     *       IN(1),ELMOUT(1),
     *       NAME,PNAME,END,BLANK8,
     *       ERRSUB,VCHANG,REALN,WRTL,WRTM,
     *       ERR1(1),ERR2(2),ERR3(2),ERR4(5),ERR5(3),ERR6(4),
     *       ERR7(6),ERR8(4),ERR10(1),
     *       ERR11(2),ERR15(2),ERR16(4),ERR17(1)
C
      INTEGER IDATA(1),
     *        LENHED,LINK,FREEH,
     *        MACROH,GLBLPH,GLBLNH,MGLPHP,LCLPH,
     *        NVLS,
     *        BTYPE,NTYPE,MTYPE,
     *        UIPTR,
     *        IPAR(24),ENAME(13),
     *        LEN,BLANK,MOUT,VNAME,
     *        IT,NMBR,PTEST,INDEX,
     *        NCHUSD,NSIGD,IER,TYPE,POSPTR,
     *        ETP,ELN,ETPP,ELNN,IWRTL,IWRTM
C
      LOGICAL*1 ERSUB1(64)
C
      EQUIVALENCE (RDATA(1),IDATA(1))
     *            ,(BLANK8,BLANK),(ERRSUB,ERSUB1)
C
      COMMON /MEMORY/ RDATA(1)
     *       /MCLIST/ VALNAM(750),LENHED(750),LINK(750),FREEH
     *       /MCHEDS/ MACROH,GLBLPH,GLBLNH,MGLPHP,LCLPH
     *       /BLCK06/ NVLS(70)
     *       /BLCK00/ BTYPE,NTYPE,MTYPE
     *       /UDELEM/ UIPTR(20)
     *       /ERRSUB/ ERRSUB(8)
C
      DATA BLANK8/' '/,LETV/'V'/,DOT/'.'/,
     *     WRTL/'WRTL'/,WRTM/'WRTM'/,END/'END'/,
     *     ERR1  /'TYPE'/,
     *     ERR2  /'NOT RECO','GNIZED'/,
     *     ERR3  /'ELEMENT/','MODEL'/,
     *     ERR4  /'INVALID ','WRTL/WRT','M SPECIF','ICATION ','FOR'/,
     *     ERR5  /'CANNOT B','E ALTERE','D'/,
     *      ERR6  /'V-VALUE ','MISSING ','OR INVAL','ID'/,
     *      ERR7  /'DUPLICAT','E ELEMEN','T/MODEL ','PARAMETE',
     *              'R NAME F','OR'/,
     *      ERR8  /'INPUT IN','FORMATIO','N TOO LO','NG'/,
     *      ERR10 /'ERROR'/,
     *      ERR11 /'MODEL/PA','RAMETER'/,
     *      ERR15 /'PARAMETE','R NAME'/,
     *      ERR16 /'MISSING ','PARAMETE','R NAME F','OR MODEL'/,
     *      ERR17 /'MODEL'/
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C...BEGIN THE LINEAR PART AND SET INITIAL CONDITION
      IF(MOUT.LT.102) GOTO 995
      ELMOUT(1)=0.0
      ELMOUT(2)=0.0
      IREF=2
      IWRTL=0
      IWRTM=0
      IETP=0
      IMDEL=0
      INDEX=1.0
      NCOUNT=1
      ICOUNT=0
      IPARM=0
      IFLAG=0
      IER1=0
      IT=LCLPH
      DO 10 I=1,8
10    ERRSUB(I)=BLANK8
      IF(IN(2).EQ.0D0) GOTO 310
C
C...FIND INPUT PARMETER 'WRTL'
20    IF(VALNAM(IT).EQ.WRTL) GOTO 30
      IT=LINK(IT)
      GOTO 20
30    IV=LENHED(IT)
      GOTO 60
50    IV=LINK(IV)
60    ICOUNT=ICOUNT+1
      IF(ICOUNT.GT.IN(2)) GOTO 220
      LEN=LENHED(IV)
      IF(LEN.GT.22) GOTO 906
      DO 65 I=1,24
65    IPAR(I)=BLANK
      DO 70 I=1,17,8
      CALL PACKCH(VALNAM(IV),IPAR(I),8,1,4)
      IF(LEN.LE.(I+7)) GOTO 80
70    IV=LINK(IV)
C
C...CHECK INPUT IS LITERAL OR NUMBER
80    CALL RDCODE(IPAR(1),LEN,NCHUSD,REALN,NSIGD,IER)
      IF(IER.EQ.1) GOTO 150
      IF(IETP.EQ.0) GOTO 901
C
C...INPUT IS A NUMBER
100   IF(NCOUNT.EQ.2) GOTO 110
      ELMOUT(IREF+3)=REALN
      NCOUNT=2
      GOTO 50
110   ELMOUT(IREF+4)=REALN
      IETP=0
      NCOUNT=1
      IFLAG=1
      GOTO 50
C
C...INPUT IS A LITERAL AND CHECK ELEMENT TYPE ETP
150   IF(IPAR(1).EQ.DOT) GOTO 901
      CALL MDPACK(IPAR,ETP,ELN,INDEX,PNAME,POSPTR,1,1,IER)
      IF(ETP.EQ.0) GOTO 901
C
C...CHECK ELEMENT LINE ELN
160   CALL MDPACK(IPAR,ETP,ELN,INDEX,PNAME,POSPTR,1,2,IER)
      IF(ELN.EQ.0) GOTO 902
      IF(ETP.EQ.7.OR.ETP.EQ.8) GOTO 170
      IF(ETP.GT.50) GOTO 170
      NMBR=NVLS(ETP)
      IF(NMBR.EQ.0) GOTO 903
C
C...CHECK THE INDEX VALUE
170   CALL MDPACK(IPAR,ETP,ELN,INDEX,PNAME,POSPTR,1,3,IER)
      IF(IER.EQ.1) GOTO 904
      IF(ETP.EQ.7.OR.ETP.EQ.8) GOTO 190
      IF(ETP.GT.50) GOTO 180
      IF(INDEX.GT.NMBR) GOTO 904
      GOTO 210
180   L=UIPTR(ETP-BTYPE)
      NMBR=IDATA(L+19)+IDATA(L+20)
      IF(NMBR.EQ.0) GOTO 903
      IF(INDEX.GT.NMBR) GOTO 904
      GOTO 210
C
C...FOR VOLTAGE OR CURRENT SOURCE
190   CALL MDSRCE(VCHANG,ELN,1,INDEX,3,IER)
      IF(IER-1) 210,903,904
C
C...BEGIN SET DATA
210   IF(IETP.EQ.1.OR.IFLAG.EQ.1) IREF=IREF+5
      ELMOUT(IREF+1)=ELN
      ELMOUT(IREF+2)=ETP
      ELMOUT(IREF+3)=0.0
      ELMOUT(IREF+4)=0.0
      ELMOUT(IREF+5)=INDEX
      IWRTL=IWRTL+1
      INDEX=1
      IETP=1
      NCOUNT=1
      GOTO 50
C
C...PREPARE OUTPUT
220   MOUT=5*IWRTL+2
      ELMOUT(1)=IWRTL
C
C...CHECK DUPLICATION ERRORS
      IF(IWRTL.EQ.1) GOTO 300
      L=MOUT-9
      N=MOUT-4
      DO 250 I=3,L,5
      K=I+5
      DO 250 J=K,N,5
      IF(ELMOUT(I+1).EQ.7D0.OR.ELMOUT(I+1).EQ.8D0) GOTO 230
      IF(ELMOUT(I+1).EQ.35D0.OR.ELMOUT(I+1).GT.50D0) GOTO 230
      IF(ELMOUT(I).EQ.ELMOUT(J)) GOTO 905
      GOTO 250
230   IF(ELMOUT(I).EQ.ELMOUT(J).AND.ELMOUT(I+4).EQ.ELMOUT(J+4)) GOTO 905
250   CONTINUE
C
C...BEGIN THE NON-LINEAR PART
300   IREF=IREF+5
      IT=LCLPH
      ICOUNT=0
      NCOUNT=1
      INDEX=1
310   IF(IN(3).EQ.0D0) GOTO 996
      IFLAG=0
C
C...CHECK PARAMETER 'WRTM'
320   IF(VALNAM(IT).EQ.WRTM) GOTO 330
      IT=LINK(IT)
      GOTO 320
330   IV=LENHED(IT)
      GOTO 360
350   IV=LINK(IV)
360   ICOUNT=ICOUNT+1
      IF(ICOUNT.GT.IN(3)) GOTO 500
      LEN=LENHED(IV)
      IF(LEN.GT.22) GOTO 906
      DO 365 I=1,24
365   IPAR(I)=BLANK
      DO 370 I=1,17,8
      CALL PACKCH(VALNAM(IV),IPAR(I),8,1,4)
      IF(LEN.LE.(I+7)) GOTO 380
370   IV=LINK(IV)
C
C...CHECK PARAMETER IF EQUAL 'END'
380   IF(VALNAM(IV).NE.END) GOTO 390
      IF(IMDEL.NE.1) GOTO 918
      IF(IPARM.NE.1.AND.IFLAG.EQ.0) GOTO 911
      IF(IPARM.EQ.1) IREF=IREF+5
      IFLAG=0
      IMDEL=0
      IPARM=0
      NCOUNT=1
      GOTO 350
C
C...CHECK INPUT IS A NUMBER OR LITERAL
390   CALL RDCODE(IPAR,LEN,NCHUSD,REALN,NSIGD,PTEST)
      IF(PTEST.NE.0) GOTO 420
C
C...INPUT IS A NUMBER
      IF(IMDEL.NE.1) GOTO 918
      IF(IPARM.NE.1) GOTO 911
      IF(NCOUNT.EQ.2) GOTO 410
      ELMOUT(IREF+3)=REALN
      NCOUNT=2
      GOTO 350
410   ELMOUT(IREF+4)=REALN
      NCOUNT=1
      IPARM=0
      IFLAG=1
      IREF=IREF+5
      GOTO 350
```

```
C...INPUT IS A LITERAL AND CHECK MODEL TYPE ETP
420   ETPP=0
      CALL MDPACK(IPAR,ETPP,ELN,INDEX,PNAME,POSPTR,1,1,IER)
      IF(ETPP.EQ.0) GOTO 450
      IF(ETPP.EQ.33.OR.ETPP.EQ.34.AND.IMDEL.EQ.1) GOTO 450
      ETP=ETPP
C
C...CHECK THE ELN
      CALL MDPACK(IPAR,ETP,ELN,INDEX,PNAME,POSPTR,2,2,IER)
      IF(ELN.EQ.0) GOTO 902
      IF(IMDEL.EQ.1) GOTO 430
      IMDEL=1
      NCOUNT=1
      GOTO 350
430   IF(IPARM.NE.1) GOTO 911
      IREF=IREF+5
      IPARM=0
      NCOUNT=1
      GOTO 350
C
C...CHECK PARAMETER
450   IF(IMDEL.EQ.0) GOTO 918
      POSPTR=0
      CALL MDPACK(IPAR,ETP,ELNN,INDEX,PNAME,POSPTR,3,2,IER)
      IF(IER.EQ.1) GOTO 914
      IF(IPAR(POSPTR).EQ.BLANK) GOTO 460
      CALL MDPACK(IPAR,ETP,ELNN,INN,PNAME,POSPTR,3,3,IER)
      IF(IER.EQ.1) IER1=1
      GOTO 470
460   INN=1
C
C...CHECK PARAMETER NAME AND INDEX VALUE
470   CALL MDMODL(VCHANG,PNAME,ELN,ETP,INDEX,INN,3,IER)
      IF(IER-1) 480,915,914
480   IF(IER1.EQ.1) GOTO 915
      IF(IMDEL.NE.1) GOTO 918
      IF(IPARM.EQ.1) IREF=IREF+5
      ELMOUT(IREF+1)=ELN
      ELMOUT(IREF+2)=ETP
      ELMOUT(IREF+3)=0D0
      ELMOUT(IREF+4)=0D0
      ELMOUT(IREF+5)=INDEX
      IWRTM=IWRTM+1
      IPARM=1
      INDEX=1
      NCOUNT=1
      GOTO 350
C
C...PREPARE THE OUTPUT
500   IF(IPARM.EQ.0.AND.PTEST.EQ.1) GOTO 917
      ELMOUT(2)=IWRTM
      MOUT=5*IWRTL+5*IWRTM+2
      IF(IWRTM.EQ.1) GOTO 996
      L=MOUT-9
      N=MOUT-4
      M=IWRTL*5+3
C
C...CHECK DUPLICATION ERRORS
      DO 510 I=M,L,5
      K=I+5
      DO 510 J=K,N,5
      IF(ELMOUT(I).EQ.ELMOUT(J).AND.ELMOUT(I+4).EQ.ELMOUT(J+4)) GOTO 916
510   CONTINUE
996   IX=IWRTL+IWRTM
      IF(IX.LE.20) GOTO 999
      ELMOUT(1)=-2
      GOTO 999
C
C...ERRORS MESSAGE
901   CALL PACKCH(ERR1,ERSUB1,4,1,1)
      CALL PACKCH(IPAR(1),ERSUB1(6),LEN,4,1)
      CALL PACKCH(ERR2,ERSUB1(LEN+7),14,1,1)
      GOTO 997
902   CALL PACKCH(ERR4,ERSUB1,35,1,1)
      CALL PACKCH(ERR3,ERSUB1(37),13,1,1)
      CALL PACKCH(IPAR(1),ERSUB1(51),LEN,4,1)
      GOTO 997
903   CALL PACKCH(ERR3,ERSUB1,7,1,1)
      CALL PACKCH(IPAR(1),ERSUB1(9),LEN,4,1)
      CALL PACKCH(ERR5,ERSUB1(LEN+10),17,1,1)
      GOTO 997
904   CALL PACKCH(IPAR(1),ERSUB1,LEN,4,1)
      CALL PACKCH(ERR6,ERSUB1(LEN+2),26,1,1)
      GOTO 997
905   ELN=ELMOUT(I)
      CALL PACKNM(ELN,ENAME,LEN,2)
925   CALL PACKCH(ERR7,ERSUB1,42,1,1)
      CALL PACKCH(ENAME,ERSUB1(44),LEN,4,1)
      GOTO 997
906   CALL USRMSG(ERR8,26,8)
      GOTO 998
911   CALL PACKCH(ERR11,ERSUB1,15,1,1)
      CALL PACKCH(IPAR(1),ERSUB1(17),LEN,4,1)
      CALL PACKCH(ERR2,ERSUB1(LEN+18),14,1,1)
      GOTO 997
914   CALL PACKCH(ERR15,ERSUB1,14,1,1)
      CALL PACKCH(IPAR(1),ERSUB1(16),LEN,4,1)
      CALL PACKCH(ERR2,ERSUB1(LEN+17),14,1,1)
      GOTO 997
915   CALL PACKCH(IPAR(1),ERSUB1,LEN,4,1)
      CALL PACKCH(ERR6,ERSUB1(LEN+2),26,1,1)
      GOTO 997
916   MNP=ELMOUT(I)-3
      CALL PACKNM(MNP,ENAME,LEN,1)
      GOTO 925
917   CALL PACKCH(ERR16,ERSUB1,32,1,1)
      CALL PACKCH(IPAR(1),ERSUB1(34),LEN,4,1)
      GOTO 997
918   CALL PACKCH(ERR17,ERSUB1,5,1,1)
      CALL PACKCH(IPAR(1),ERSUB1(7),LEN,4,1)
      CALL PACKCH(ERR2,ERSUB1(LEN+8),14,1,1)
997   CALL ERRMSG(108159)
998   ELMOUT(1)=-1
      MOUT=1
      GOTO 999
995   MOUT=0
999   RETURN
      END
```

```
C***********************************************************************
      SUBROUTINE MDRUSR(IREP,PAR)
C***********************************************************************
C
C     MCDC ANALYSIS SUBROUTINE
C     RUNS BEFORE DC ANALYSIS TO CHANGE ELEMENT VALUE
C
C     IREP = NUMBER OF THE CURRENT #RUSER REPEAT
C     PAR  = ELEMENT/PARAMETER INFO
C     INPFL ^= 1  TO STOP DISPLAY OF INPUT INFORMATION
C     ICOUNT= # OF CURRENT CHANGED ELEMENT
C
C     CREATED 03-AUG-1987     J.F. SUEN  EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*4 TIME
      REAL*8 RDATA,
     *       LVLID,PRIVID,
     *       DATE,
     *       WDJTID,
     *       DSEED,PARM,PNAME,RANVAL,PAR1,PAR2,DABS,
     *       DSQRT,INIVL,VCHANG,TEMP,FNAME,
     *       PAR(1),FILEID(3),FILEIP(3),TITLE(4),
     *       INITL(20),AVG(20),SDEV(20),LARG(20),SMAL(20),
     *       SUM(20),SUMW(20),BUFFER(10),RNAME(3)
C
      INTEGER IDATA(1),
     *        NELEMS,NNODES,SIPTR,SRPTR,ETPTR,
     *        RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW,
     *        UIPTR,
     *        BTYPE,NTYPE,MTYPE,
     *        EDITID,
     *        WDJTFL,
     *        ATTCD,
     *        ANALYZ,
     *        EDIDA,MDIPTR,MDRPTR,NDIM,NPAR,NFIXLP,
     *        INPFL,OUCODE,MIDAPT,IREP,ITEMP,APNTR,KEPTR,
     *        IER,RPTR,NMBR,ERR,SWTYP,
     *        INN,NS,MNP,P,ELN,ETP,INDEX,BLANK,LETV,DOT,
     *        IPARM(8),NAME(24),
     *        DIGIT(10),NUMB(10)
C
      LOGICAL*1 NAME1(24)
C
      EQUIVALENCE (RDATA(1),IDATA(1)),(RNAME,NAME1)
C
      COMMON /MEMORY/ RDATA(1)
     *       /CPNTRS/ NELEMS,NNODES,SIPTR,SRPTR,ETPTR
     *       /BINPUT/ RDUNT,WRUNT,TRMUNT,WRTSW,TRMSW,RDSW,EOFSW
     *       /UDELEM/ UIPTR(20)
     *       /BLCK00/ BTYPE,NTYPE,MTYPE
     *       /BLCK07/ LVLID,PRIVID,EDITID
     *       /DATEBL/ DATE(3)
     *       /WDJTBL/ WDJTID,WDJTFL
     *       /FNBLOK/ FNAME
     *       /ATTIBL/ ATTCD
     *       /APNTRS/ ANALYZ(30)
C
      DATA FILEID/'MCDC','MDKPINFO','A'/,
     *     FILEIP/'MCDC','MDKPVALS','A'/,
     *     TITLE/4*' '/,
     *     NUMB /0,1,2,3,4,5,6,7,8,9/,
     *     DIGIT /'0','1','2','3','4','5','6','7','8','9'/,
     *     BLANK/' '/, LETV/'V'/, DOT/'.'/
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IF(IREP.GT.1) GOTO 50
C
C...READ INPUT DATA
      OUCODE=PAR(1)
      NS=PAR(5)
      DSEED=PAR(6)
      IWRTL=PAR(7)
      IWRTM=PAR(8)
      ITOTAL=IWRTM+IWRTL
      INPFL=1
      IF(DSEED.NE.0D0) GOTO 100
C
C...PICK DSEED NUMBER FROM CPU TIME
      CALL GTIMER(TIME)
      TEMP=10.0*TIME
      ITEMP=TEMP
      TEMP=TEMP-ITEMP
      TEMP=TEMP*10D8
      ITEMP=TEMP
      DSEED=ITEMP
      GOTO 100
50    IF(IREP.GT.NS) GOTO 300
C
C...BEGIN DECODING THE INPUT DATA
100   INCRSE=0
      DO 250 I=1,ITOTAL
      ELN=PAR(I+INCRSE+8)
      ETP=PAR(I+INCRSE+9)
      PAR1=PAR(I+INCRSE+10)
      PAR2=PAR(I+INCRSE+11)
      INDEX=PAR(I+INCRSE+12)
      IF(IREP.GT.1) GOTO 160
C
C...FIRST COMING IN SET INITIAL VALUE AND FIND MEAN VALUE
      IF(I.GT.IWRTL) GOTO 120
      IF(ETP.EQ.7.OR.ETP.EQ.8) GOTO 110
      RPTR=IDATA(SRPTR+ELN)-1
      INITL(I)=RDATA(RPTR+INDEX)
      GOTO 150
C
C...VOLTAGE OR CURRENT SOURCE
110   CALL MDSRCE(VCHANG,ELN,1,INDEX,1,IER)
      INITL(I)=VCHANG
      GOTO 150
C
C...CHANGE MODEL VALUES
120   CALL MDMODL(VCHANG,PNAME,ELN,ETP,INDEX,INN,1,IER)
      INITL(I)=VCHANG
C
C...SET INITIAL CONDITION
150   SUM(I)=0D0
      SUMW(I)=0D0
      AVG(I)=0D0
      SDEV(I)=0D0
```

```
C
C...CALL RANDOM NUMBER AND SORTING
160   INIVL=INITL(I)
      IER=0
      ICOUNT=I
      CALL MDRAND(RANVAL,TITLE,INPFL,IREP,ICOUNT,DSEED,INIVL,
     *            PAR1,PAR2,IER)
      IF(IER.NE.0) GOTO 997
      SUM(I)=SUM(I)+RANVAL
      SUMW(I)=SUMW(I)+RANVAL**2
      IF(IREP.GT.1) GOTO 170
      LARG(I)=RANVAL
      SMAL(I)=RANVAL
      GOTO 180
170   IF(RANVAL.GT.LARG(I)) LARG(I)=RANVAL
      IF(RANVAL.LT.SMAL(I)) SMAL(I)=RANVAL
C
C...CHANGE VALUES (LINEAR AND MODEL)
180   IF(I.GT.IWRTL) GOTO 220
      IF(ETP.EQ.7.OR.ETP.EQ.8) GOTO 210
      RPTR=IDATA(SRPTR+ELN)-1
      RDATA(RPTR+INDEX)=RANVAL
      GOTO 250
C
C...CHANGE VOLTAGE OR CURRENT SOURCE
210   CALL MDSRCE(RANVAL,ELN,1,INDEX,2,IER)
      GOTO 250
C
C...CHANGE MODEL PARAMETER VALUE
220   CALL MDMODL(RANVAL,PNAME,ELN,ETP,INDEX,INN,2,IER)
250   INCRSE=INCRSE+4
      GOTO 999
C
C...FIND FILE NAME AND FILE TYPE
300   APNTR=ANALYZ(1)
      KEPTR=IDATA(APNTR+35)
      FILEID(1)=RDATA(IDATA(KEPTR))
      FILEIP(1)=RDATA(IDATA(KEPTR))
      FILEID(3)=RDATA(IDATA(KEPTR)+1)
      FILEIP(3)=RDATA(IDATA(KEPTR)+1)
C
C...PREPARE OUTPUTS
370   CALL IOOPEN(FILEID,0,10*8,1,IER)
      IF(IER.NE.0) GOTO 997
      BUFFER(1)=ITOTAL
      BUFFER(2)=NS
      BUFFER(3)=OUCODE
      BUFFER(4)=TITLE(1)
      BUFFER(5)=TITLE(2)
      BUFFER(6)=TITLE(3)
      BUFFER(7)=TITLE(4)
      BUFFER(8)=DATE(1)
      BUFFER(9)=DATE(2)
      BUFFER(10)=DATE(3)
      CALL IOWRIT(FILEID,BUFFER,10*8,IER)
      IF(IER.NE.0) GOTO 997
      CALL IOCLOS(FILEID,0,IER)
      IF(IER.NE.0) GOTO 997
      IF(INPFL.NE.1) GOTO 380
      WRITE(WRUNT,1300)LVLID,EDITID,PRIVID,DATE,WDJTID,FNAME
      WRITE(WRUNT,1400)NS,TITLE(1),TITLE(2),TITLE(3),TITLE(4)
      WRITE(WRUNT,1200)
C
C...OPEN DISK FILES
380   CALL IOOPEN(FILEIP,0,8*8,1,IER)
      IF(IER.NE.0) GOTO 997
      INCRSE=0
      DO 750 I=1,ITOTAL
      ELN=PAR(I+INCRSE+8)
      ETP=PAR(I+INCRSE+9)
      INDEX=PAR(I+INCRSE+12)
C
C...CALCULATE STATISTICAL DATA
      AVG(I)=SUM(I)/NS
      SDEV(I)=SUMW(I)/NS-AVG(I)**2
      SDEV(I)=DSQRT(SDEV(I))
C
C...PUT INITIAL VALUE BACK TO ELEMENT
      IF(I.GT.IWRTL) GOTO 395
      IF(ETP.EQ.7.OR.ETP.EQ.8) GOTO 390
      RPTR=IDATA(SRPTR+ELN)-1
      RDATA(RPTR+INDEX)=INITL(I)
      GOTO 500
C
C...FOR VOLTAGE AND CURRENT SOURCE
390   VCHANG=INITL(I)
      CALL MDSRCE(VCHANG,ELN,1,INDEX,2,IER)
      GOTO 500
C
C...FOR NON-LINEAR ELEMENT
395   VCHANG=INITL(I)
      CALL MDMODL(VCHANG,PNAME,ELN,ETP,INDEX,INN,2,IER)
C
C...DISPLAY THE OUTPUT AND STORE STATISTICAL INPUT VALUES IN DISK
500   DO 520 J=1,24
520   NAME(J)=BLANK
      IF(I.GT.IWRTL) GOTO 550
      CALL PACKNM(ELN,NAME,II,2)
      INN=INDEX
      IF(INDEX.EQ.1) GOTO 700
      NAME(II+1)=DOT
      NAME(II+2)=LETV
      L=II+2
      GOTO 600
C
C...FIND THE QUALIFIER VALUES
550   MNP=ELN-3
      CALL PACKNM(MNP,NAME,II,1)
      CALL MDMODL(RANVAL,PNAME,ELN,ETP,INDEX,INN,4,IER)
      CALL PACKCH(PNAME,IPARM(1),8,1,4)
      DO 560 II=1,6
      IF(IPARM(II).EQ.BLANK) GOTO 570
560   NAME(II+14)=IPARM(II)
570   IF(INN.EQ.1) GOTO 700
      NAME(II+14)=DOT
      NAME(II+15)=LETV
      L=II+15
C
C...DECODE THE INDEX NUMBER
600   DO 610 K=1,6
```

```
      INUMB=INN/10**(6-K)
      IF(INUMB.NE.0) GOTO 620
610   CONTINUE
620   K=7-K
      TEMP=INN
      DO 650 J=1,K
      TEMP=TEMP/10**(K-J)
      INUMB=TEMP
      TEMP=(TEMP-INUMB)*10**(K-1)
      L=L+1
      DO 630 N=1,10
      IF(INUMB.EQ.NUMB(N)) GOTO 650
630   CONTINUE
650   NAME(L)=DIGIT(N)
700   IF(INPFL.NE.1) GOTO 720
C
C...WRITE OUTPUT
      WRITE(WRUNT,1100)(NAME(J),J=1,24),INITL(I),SMAL(I),
     *                 LARG(I),AVG(I),SDEV(I)
C
C...WRITE DATA IN DISK
720   CALL PACKCH(NAME,NAME1(1),24,4,1)
      BUFFER(1)=RNAME(1)
      BUFFER(2)=RNAME(2)
      BUFFER(3)=RNAME(3)
      BUFFER(4)=INITL(I)
      BUFFER(5)=SMAL(I)
      BUFFER(6)=LARG(I)
      BUFFER(7)=AVG(I)
      BUFFER(8)=SDEV(I)
      CALL IOWRIT(FILEIP,BUFFER,8*8,IER)
      IF(IER.NE.0) GOTO 997
750   INCRSE=INCRSE+4
C
C...CLOSE THE DISK
      CALL IOCLOS(FILEIP,0,IER)
      IF(IER.NE.0) GOTO 997
      GOTO 999
C
C...ERROR
997   ATTCD=1
999   RETURN
1100  FORMAT(1X,24A1,1P5E11.3)
1200  FORMAT(' INPUT                       INITIAL    MINIMUM    MAXIMUM
     *    MEAN       ST.DEV'/)
1300  FORMAT(/1X,'MCDC',1X,A6,A2,1X,A8,1X,2A8,A4,A8,'  FILE: ',A8)
1400  FORMAT(/5X,'NUMBER OF SAMPLES = ',I4,9X,4A8/)
      END
```

```
C***********************************************************************
      SUBROUTINE MDSRCE(SDATA,ELN,CODE,INDEX,TYPE,IER)
C***********************************************************************
C
C     MCDC ANALYSIS ROUTINE
C     ROUTINE DECODES INDEPENDENT SOURCE SPECIFICATION
C
C     TYPE = 1 FIND INITIAL VALUE FOR SOURCE
C          = 2 CHANGE SOURCE VALUE BY USING RANDOM VALUE
C          = 3 FIND QUALIFIER VALUE FOR SOURCE
C     IER = 0 NO ERROR
C           1 SOURCE NOT RECOGNIZED
C           2 V-VALUE MISSING OR INVALID
C     SDATA = SOURCE INITIAL VALUE/RANDOM VALUE
C
C     CREATED  03-AUG-1987     J.F. SUEN  EE DEP.  YSU
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      REAL*8 RDATA,
     *       PERIOD,NAME,SDATA,R1
C
      INTEGER IDATA(1),
     *        NELEMS,NNODES,SIPTR,SRPTR,ETPTR,
     *        IPTR,INDEX,STYPE,NARGS,STIPTR,
     *        ELN,CODE,TYPE,INDEX,IER
C
      EQUIVALENCE (RDATA(1),IDATA(1))
C
      COMMON /MEMORY/ RDATA(1)
     *       /CPNTRS/ NELEMS,NNODES,SIPTR,SRPTR,ETPTR
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C...SET INITIAL CONDITION
      IER=0
      I=IDATA(ETPTR+ELN)
      IPTR=IDATA(SIPTR+ELN)+36-4*I
      STYPE=IDATA(IPTR)
      STIPTR=IDATA(IPTR+3)
      NARGS=IDATA(IPTR+1)
      IF(INDEX.GT.NARGS) GOTO 901
      IF(STYPE.EQ.5.OR.STYPE.EQ.6) GOTO 902
      IF(TYPE.EQ.3) GOTO 999
      N=IDATA(STIPTR+3)-1
C
C...FIND INITIAL VALUE
      IF(TYPE.EQ.2) GOTO 100
      SDATA=RDATA(N+INDEX)
      GOTO 999
C
C...CHANGE VALUE
100   RDATA(N+INDEX)=SDATA
      GOTO 999
C
C...ERRORS
901   IER=2
      GOTO 999
902   IER=1
999   RETURN
      END
```

LIST OF REFERENCES

1. Nagel,L.W., SPICE2: A Computer Program to Simulate Semiconductor Circuits, U. of California, Berkeley, 1975.

2. Bryant,P.R., Fingerote,S., Hajj,I.N., Skelbae,S. and Vlach,M., WATAND Primer (Version 1.09), U. of Waterloo, November 27, 1980.

3. Hammersley,J.M. and Handscomb,D.C., Monte Carlo Methods, Ballantyne & Co. Ltd., Great Britain, 1964.

4. Lewis,T.G. and Smith,B.J., Computer Principles of Modeling and Simulation, Houghton Mifflin Inc., 1979.

5. Gordon,G., System Simulation, IBM Corporation New York Scientific Center, Prentice-Hall Inc., 1978.

6. Kennedy,J.B. and Neville,A.M., Basic Statistical Methods for Engineer and Scientists, Harper & Row, Inc., 1986.

7. Vlach,M., WATAND User's Manual (Version 1.09-08), revised by Bryant,P.R. and Strayer,H.J., U. of Waterloo, December 15, 1985.

8. Hajj,I.N., Singhal,K., Vlach,J. and Bryant,P.R., A Program for the Analysis and Design of Linear and Piecewise-Linear Networks, Proceedings of the Sixteenth Midwest Symposium on Circuit Theory, U. of Waterloo, April 12-13, 1973.

9. Munro,P.C., WATAND Help Files on YSU Computer System, 1985.

10. Munro,P.C., "CHAPTER 9 Imbed and Macro Facilities", rewritten for WATAND User's Manual, 1987.

11. Munro,P.C., "Section 4.19 Repeating An Analysis with #RUSER", addition for WATAND User's Manual, 1986.

12. Gottfried,B.S., Elements of Stochastic Progress Simulation, Prentice-Hall Inc., 1984.

13. Fishman,G.S., Concepts and Methods in Discrete Event Digital Simulation, John Wiley & Sons, Inc., 1973.

14. IMSL Library, Edition 9.2, Revised November, 1984.