DOE/ET/29196--T2

DOE/ET/29196--T2

DE83 017943

Final Report
on
DOE Contract No. EC-7-S-01-5102
An Efficient Technique for Calculating
Multiple Solutions
of Electric-Field Problems.
Part II

Principal Investigator: T. S. Lauber
S. B. Crary Professor of Engineering

Rensselaer Polytechnic Institute
Troy, New York
May 1981

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

NOTICE
PORTIONS OF THIS REPORT ARE ILLEGIBLE.

It has been reproduced from the best available copy to permit the broadest possible availability.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Foreword to Part II

The computer program described herein was written by George G. Theodossiou in Fortran IV for use on the IBM 3033 system at RPI. It was designed to be as portable as possible.

The project was to modify an existing program which had been written by Carlos E. Varela for use on the Prime 500 system at RPI. The original program is described in Part I.

Prospective users of either program should be familiar with the contents of both parts.

Finally, it should be noted that Mr. Theodossiou was not supported by funds from this contract. Only computer run costs were borne by the contract.

Manual Cycled separately

Corrections and modifications
of the "WINDOW" program

A project submitted by George G. Theodossiou
to
Rensselaer Polytechnic Institute
in partial fulfillment of requirements
for the degree of
Master of Engineering
in
Electric Power Engineering

Troy, New York
April 29, 1981

APPROVED:

T. S. Lauber 8 May, 1981

T. S. Lauber

S. B. Crary Professor of Engineering

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

CONTENTS

	Page
LIST OF TABLES	iv
LIST OF FIGURES	iv
ACKNOWLEDGEMENT	v
1. INTRODUCTION	1
2. GENERAL	6
3. THE PROGRAM	9
4. DESCRIPTION OF ALTERATIONS	12
4.1 Alterations in MAIN	13
4.2 Alterations in CODING	16
4.3 Alterations in EQUIVALENT	26
4.4 Alterations in SOLUTN	31
4.5 Alterations in PLOTY	35
5. THE NEW SUBROUTINE "PLOT1"	44
6. CONCLUSIONS	48
OPERATOR'S MANUAL	follows

LIST OF TABLES

	Page
Table 3.1 The subroutines of the program	11
Table 4.1 Alterations in Main program	14
Table 4.2 Alterations in CODING	17
Table 4.3 Alterations in EQUIVALENT	28
Table 4.4 Alterations in SOLUTN	32
Table 4.5 Alterations in PLOTY	36

LIST OF FIGURES

	Page
Figure 4.1 Flow-chart of new MAIN program	15
Figure 5.1 Flow-chart of new subroutine PLOT1	46

ACKNOWLEDGEMENT

I want to thank very much my adviser, Dr. Lauber, for his continuous cooperation and for all the hours which he spent helping me to get through with the project.

I also want to express my deepest gratitude to my parents for the support which they gave me during the years of my studies, and especially during this year of studies at R.P.I. , from which studies I gained so much knowledge. Without them I would have never been able to get this invaluable experience.

Finally, a special thanks goes to my wife Marina, who typed this paper and whose precious help was always there. This paper is for her.

George Theodossiou

1. INTRODUCTION

One of the areas of work presently under progress at Electric Power Engineering Department of Rensselaer Polytechnic Institute is the calculation of electric and magnetic fields with the aid of digital computer programs. A number of programs made to calculate fields by employing various methods have been built in Rensselaer, and there are more under construction now. Projects for two-dimensional as well as three-dimensional electric and magnetic fields are progressing.

One of these projects is the present which I took as a 3-credit Electric Power Engineering Project for the Fall semester of 1980. The subject is a program written in FORTRAN IV which calculates electric fields with two-dimensional flat geometry or three-dimensional geometry having axial symmetry. This program originates from a field program written by Andersen. Since the time it was written there have been made many changes by various students which got involved with it. New parts were introduced to increase its capability, its efficiency and to reduce the cost of running. The last one to get involved with the program was student Carlos E. Varela who made major changes and left the "WINDOW" version of it, a very economic and effi-

cient version. The program was left to reside in the PRIME computer system, loaded into file "WINDOW".

My purpose, when I started this project, was to transfer the program from the PRIME to the IBM computer system. This would require some changes to be made, so that the program would become compatible with the peculiarities of the IBM machine. It all seemed to be an easy thing to be accomplished and the agreement between Dr. Lauber and me was that after this would have been done, I would spend the rest of the semester trying to improve the efficiency of the program by changing some parts of it.

The whole thing, however, turned out not to be that easy. I spend approximately two months to debug and understand the program and I finally loaded it to the IBM computer. I made the first efforts to run the example which Mr Varela ran on the PRIME system with success, but it wouldn't work. There were many errors in it which had to be traced and corrected. These errors were not traced by the PRIME system because of its special features which would tolerate them. For example, many of the variables and arrays in the various subroutines were not initiated. This wouldn't cause anything erroneous on the PRIME system, because its operating system would initiate all variables and arrays before starting the execution of the program, but it generated numerous malfunctions such as infinite

loops etc. during execution on the IBM system, which doesn't have this initializing feature.

The first semester was over, and I only had succeeded in clearing all the errors which occurred during the compilation stage, and a few of them occurring during the execution of the program, but it wasn't still running. I took it again during the Spring 1981 semester as an Electric Power Engineering Project with three more credits. After using the very efficient (but expensive) Interactive FORTRAN (IF) compiler, I was able to make that first example run with success.

Since then, I ran a number of examples and every time I tried to run one of them, I was running into more program errors. I worked hard to be able to locate them all and correct them, and I did that, but time was running and I didn't have enough of it left to get into the program and change its algorithm to make it even more efficient. We agreed with Dr. Lauber to leave this and concentrate our efforts into giving a perfectly operating program with no errors in it, and I am confident that we succeeded in that. We also agreed that until the end of the semester I should improve the input procedure by writing a new subroutine which would give the future users the ability to check the data they give by means of graphic outputs, and to stop the execution when something was going wrong. This was

done, and the new subroutine PLOT1 took its place into the program.

Part of my obligations, also, was to write a User's Manual for the program. Mr. Varela had already written one as part of his thesis, but it was not very explicit, so we decided to prepare the new one explaining the program to the user in detail. This was done, and the manual which resulted is bound here after the end of this report. The fact is that it is written to be a completely separate volume, and all its structure is made according to this idea; even the numbering of the pages in it starts from 1. It has its own lists of contents, figures and tables and starts with its own introduction. It doesn't have to do anything with the report, but, as it is part of the whole work it is submitted here, cobound with it.

Another point of the whole thing was that future users should be provided with directions to change the subroutines which perform the graphic outputs and adapt them to the graphics packages which they possess. These instructions, concerning more programmers than users, are given as an appendix in the Operator's Manual. They are given in such a way that one doesn't need to go through the logic of the whole subroutine, but he only has to change a small number of graphics subroutines which are specified exactly, and substitute them with subroutines from the

graphics package which he has in his software. We hope that the instructions given will be adequate for an easy operation of the program, although its use might seem complicated initially because of the number of different cases which it can handle and the many options that offers to the user.

2. GENERAL

As explained before, the project work includes the correction of the various errors and incompatibilities between the program and the IBM system, the creation of a new graphics subroutine which was added to the program, and the preparation of a new detailed Operator's Manual. The purpose of this paper is to give a detailed report of the work that has been done, for all three parts of the project which have been stated above.

The report of the first part of the project will include a list of all the changes that have been made to the program since I started working on it. The two editions which are compared are the program as it was left into the PRIME system and the program as it exists in the IBM system. The comparison between these two editions was done by program *APC of the Michigan Terminal System (MTS) used by the IBM computer, and it gave a list of all the changes that have been made. The outputs of *APC are included in this report. Based on them, there will be explained for each change the reason for which it was made, the way the program worked before the change and the way it works after it has been changed.

The report of the second part of the project, which

is the new subroutine PLOT1, will include the reasons for which this subroutine was written, the organization, the flow chart and a few things about the way it works.

Finally, the third part of the project, which is the OPERATOR'S MANUAL will come at the end of the report. As it is self explanatory as far as the work done for it is concerned, there will not be a separate chapter about it in the report. The only thing that we can say here is that the trend was to make it as explicitly written and detailed as possible. It is true that no matter how complicated and effectively built a program is, most of the future users will judge it from the I/O procedures, because this is the only part of the program with which they come into straight contact. An efficient and easy data-input procedure as well as a nice and explicit print- or plot-output will gain a lot of good comments for the program, while a difficult and tiring data-input procedure or a print-output with numerous numbers printed but with only a few comments printed as titles or explanations will lead to a bad criticism, although the program might be very efficient and difficult to be built. Consequently, a good and detailed Operator's Manual will ease the users with the data-input procedure and help them to increase the program's efficiency as a valuable tool for easy and quick anticipation of problems related to electric fields.

As far as the Programmer's Manual is concerned, Mr. Varela has left a very good one as part of his thesis. This, combined with the chapter of this report which describes the new subroutine PLOT1, will guide other programmers which will attempt to make changes or additions to the program in the future.

3. THE PROGRAM

We shall write here a few things about the program. The way it works can be described briefly as follows: It reads the data which describe a configuration of electric components (electrodes, dielectrics etc.) and applies a normal grid on it to divide it into finite elements. This is done by subroutine CODING. The electric field is to be calculated and plotted for a subregion of the whole area of the configuration called "window" and specified by the user. The first run of the program has the purpose to convert the area outside the window into an equivalent of it fitting on the window boundaries. This, is done by subroutine EQUIVALENT. Then, having calculated this equivalent the user can describe various alternatives of the configuration inside the window and get the resulting electric field for each one of them at relatively low cost, compared with the cost of running the program for the whole area for each alternative. The computation of the electric field inside the window is done by subroutine SOLUTION and the resulting equipotential lines are plotted by subroutine PLCTY. Another subroutine giving graphical output is PLOT1 which is the one added recently and which gives plots of the configuration with the applied grid on it.

There are also various other small subroutines and functions which aid the operation of the major subroutines referred above and which are listed in table 3.1 .

The execution of the program and the cooperation of its subroutines are coordinated by a small MAIN program, which controls the whole procedure.

Having all the information given in this chapter in mind we can now proceed to the description of the project work which was done.

MAIN Main program.

SUBROUTINES

CODING	Input subroutine.
EQUIVALENT	Calculates the equivalent of the region outside the window.
SOLUTION	Calculates the electric field inside the window.
PLOTY	Plots the resulting electric field.
PLOT1	Plots the configuration with the applied grid on it.
FSTR	Calculates field strength into dielectric materials.
PERTRI	It handles storing of the finite triangles in a proper way.
SUBD	Computes the area of a triangle.
INTER	Determines whether a point is internal to a closed contour.
DIGIT	Reads coordinates of input points and generates preprogrammed shapes.

FUNCTIONS

NODFT	Handles position of elements in symmetrical linear arrays.
NBOUFT	Determines if a point is on the boundary of the window.

Table 3.1

4. DESCRIPTION OF ALTERATIONS

Alterations were made throughout the program, and they can be divided into three main categories:

- The first category includes the modifications made in order to ammend the incompatibilities between IBM 3033 and PRIME computer systems, especially in special instructions for file manipulation, I/O operations etc.
- The second category includes the corrections of all the errors which were discovered in the program.
- The third category includes alterations made with the purpose to change the algorithm which was used in certain places in the program.

From all the subroutines listed in table 3.1 only a few of them have alterations, and these are MAIN, CODING, SOLUTN, EQUIVALENT, and PLOTY. Most of the changes were made in subroutines CODING and PLOTY. For each subroutine where alterations were made, we shall give a table which lists all the differences between Mr. Varela's initial edition of the program and the final edition which resulted from the project work. This table is the one printed by the public program *APC which was used to compare the two edi-

tions. For each difference, we shall give details of the corresponding change, as described in chapter 2 of this report.

We begin now with the MAIN program.

4.1 Alterations in MAIN. The alterations which were made in MAIN program are listed in table 4.1. The way the old program was written would make it continue the execution if an error due to erroneous input occurred in subroutine CODING. This of course was not desirable, and for this reason the variable WARN, which monitors the error status in subroutine CODING, was made to communicate with the MAIN program through the common block BLOK33 which was introduced and inform it about the error status. If an error has occurred, the program holds execution and prints the message:

***** JOB ABORTED *****

The new flow-chart corresponding to the MAIN program is the one of figure 4.1 .

The above correspond to all alterations listed in table 4.1 but the first two. In the old edition the array IVERT was declared as an INTEGER*2 array in all subroutines (passing to them through the common block BLOK11) except MAIN. This was traced by the IBM 3033 FORTRAN IV compiler, and the array with its dimensions are declared in line 11

```

Unit 0: -CID lnr 1 len 43 col 1
*****
11      INTEGER*2 NVERT(70,100)
*****
19      CCKPCN/BLOCK11/ITRT(900,F), IVERT(2000,3)
*****
After line 21
*****
23      IF(EQLIN.NE.0.)GO TO 10
*****

*****
After line 28
*****

```

```

Unit 1: -NEW lnr 1 len 43 col 1
10 equal lines *****
11      INTEGER*2 NVERT(70,100), IVERT(2000,3)
12      INTEGER WARN
7 equal lines *****
20      COMMON/BLOCK11/ITRT(900,R), IVEST
2 equal lines *****
23      COMMON/BLOCK33/IGRD, WARN
1 equal line *****
25      IF (WARN.EQ.0.OR.WARN.EQ.55) GOTO 5
26      WRITE (6,100)
27      100  FORMAT (/5X,'***** JOB ABORTED *****')
28      STOP
29      5    IF(EQLIN.NE.0.)GO TO 10
5 equal lines *****
35      STOP
1 equal line *****

```

Table 4.1

Figure 4.1

of the new edition as INTEGER*2, while its name only and not the previously specified dimensions is included in the contents of BLOK11 (line 22 of the new edition).

4.2 Alterations in CODING. The alterations which were made in subroutine Coding are listed in table 4.2 . We shall examine them one by one.

- Line 2 of the old edition was deleted. This was a special file-manipulating instruction of PRIME system which had no reason to exist in the IBM 3033 system.

- The name and dimensions of array MVERT were deleted from line 7 of the old edition and were declared separately in the new edition (line 5) as an INTEGER*2 array. This was done because although MVERT was declared as an INTEGER*2 array in the MAIN program, it was declared as a simple integer array in CODING. This was accepted by the compiler of the PRIME system, but not by the compiler of the IBM 3033 system.

- A new line (15) was introduced to the new edition. This line specifies the contents of the common block BLOK33, through which variable WARN, which monitors the error status, communicates with the MAIN program.

- Lines 23 and 25 of the old program had a spelling error. The word SYMMETRI in both of them was corrected to SYMMETRY (corresponding lines: 24 and 26 in the new program).

- Line 36 in the old program or 37 in the new program: The FORMAT which prints 'ERRONEOUS INPUT' if we have an error, was changed to print also an integer (I2) code which specifies the type of error which has occurred.

Table 4.2

Unit 0: -CIC lnr 1 len 24 col 1	Unit 1: -NEW lnr 1 len 24 col 1
1 equal line	1 equal line
2 INSERT S1SCCM>KEYC.F	After line 1
3 equal lines	3 equal lines
After line 5	5 INTEGER*2 NVERT(70,100)
1 equal line	1 equal line
7 INTRI(8),AUX(500,2),ILN(70),NZV(15),NZH(15),NVERT	7 INTRI(8),AUX(500,2),ILN(70),NZV(15),NZH(15)
170,100)	
7 equal lines	7 equal lines
After line 14	15 COMMON/BLOCK33/YGRD,WARN
8 equal lines	8 equal lines
23 123F AXIS OF SYMMETRY:- X -)	24 123R AXIS OF SYMMETRY:- X -)
1 equal line	1 equal line
25 123H AXIS OF SYMMETRY:- Y -)	26 123H AXIS OF SYMMETRY:- Y -)
10 equal lines	10 equal lines
36 45 PCFMT(16HORRORRONEOUS INPUT)	37 45 PCFMT(16HORRORRONEOUS INPUT,IX,IJ)
26 equal lines	26 equal lines
63 C --- OPEN INPUT FILE "WINDAT"	After line 63
64 CALL SFCHS(KSREAL,'WINDAT',6,1,0,NCOD)	
65 C --- OPEN COTPUT FILE "WINGUT"	
66 CALL SFCHS(KSWRITE,'WINGUT',6,2,0,NCOD)	
75 1152 REAL(IN,2) PCHV	
8 equal lines	8 equal lines
159 116 IF(K.GT.LNVER) K=LNVER	72 1152 READ (IN,9369) YGRD1,YGRD2,YGRD3
168 118 IF(K.GT.LNHCR) K=LNHCR	73 9369 FORMAT (3F10.0)
196 3100 WRITE(OUT,76)	74 READ(IN,2) PCHV
235 IF (E1.GT.TOLR) GO TO 145	83 equal lines
After line 237	158 116 IF(K.GT.LNVER.OR.LNVER.LT.10) K=LNVER
8 equal lines	8 equal lines
247 IF (E1.GT.TOLF) GO TO 147	167 118 IF(K.GT.LNHCR.OR.LNHCR.LT.10) K=LNHCR
After line 249	27 equal lines
349 1152 REAL(IN,2) PCHV	195 3100 WARN=1
349 1152 REAL(IN,2) PCHV	196 WRITE(OUT,76)
349 1152 REAL(IN,2) PCHV	38 equal lines
349 1152 REAL(IN,2) PCHV	235 IF (E1.GT.TOLR.OR.E1.EQ.0.) GO TO 145
349 1152 REAL(IN,2) PCHV	2 equal lines
349 1152 REAL(IN,2) PCHV	238 WRITE (OUT,60) LINAC,I,N
349 1152 REAL(IN,2) PCHV	239 60 FORMAT ('O LINE ',I2,' : POINT ',I3,' HAS BEEN MO
349 1152 REAL(IN,2) PCHV	VED TO POINT '
349 1152 REAL(IN,2) PCHV	240 1,I3/' WHICH WAS TOO CLOSE')
349 1152 REAL(IN,2) PCHV	8 equal lines
349 1152 REAL(IN,2) PCHV	249 NM=I-ILN(J-1)
349 1152 REAL(IN,2) PCHV	1 equal line
349 1152 REAL(IN,2) PCHV	251 IF (E1.GT.TOLR.OR.E1.EQ.0.) GO TO 147
349 1152 REAL(IN,2) PCHV	2 equal lines
349 1152 REAL(IN,2) PCHV	254 WRITE (OUT,61) L,LINAC,M,I
349 1152 REAL(IN,2) PCHV	255 61 FORMAT ('O POINT ',I3,' ON CONTOUR ',I2,' HAS BEP
349 1152 REAL(IN,2) PCHV	N MOVED TO',
349 1152 REAL(IN,2) PCHV	256 1' POINT ',I3,' ON CONTOUR ',I2/' WHICH WAS TOO C
349 1152 REAL(IN,2) PCHV	LOSE')
349 1152 REAL(IN,2) PCHV	92 equal lines
349 1152 REAL(IN,2) PCHV	349 IGMT3=0
349 1152 REAL(IN,2) PCHV	350 IF (YGRD1.EQ.0.) GOTO 4141
349 1152 REAL(IN,2) PCHV	351 JP=0
349 1152 REAL(IN,2) PCHV	352 CALL PLOT1 (YGRD1,RMTX,RMTY)
349 1152 REAL(IN,2) PCHV	353 4141 IF (YGRD1.IT.0.) IGMT3=1
349 1152 REAL(IN,2) PCHV	104 equal lines
349 1152 REAL(IN,2) PCHV	458 1667 ASINH=100000.
349 1152 REAL(IN,2) PCHV	53 equal lines
349 1152 REAL(IN,2) PCHV	512 IF (K.EQ.0) GOTO 999
349 1152 REAL(IN,2) PCHV	10 equal lines
349 1152 REAL(IN,2) PCHV	

510	1720	IF (JS.EQ.0) GO TO 1721	523	1720	IF (JS.EQ.0) GO TO 1721
514	1721	ASSA1=ABS (ASINB*SALPH)	527	1721	IF (ABS (ASINB*SALPH).GT.0.01.OR.ABS (ASTHR*CALPH).
515		ASCII=ABS (ASINB*CALPH)			
516		IF (ABS (ASINB*CALPH).GT.0.01.OR.ABS (ASTHR*CALPH).			
After line 560			44 equal lines	After line 560	
After line 567			7 equal lines	After line 567	
571		IF (N2.NE.1) GO TO 184	3 equal lines	580	IF (N2.NE.1) GO TO 185
598	C ---	APCIT	26 equal lines	611	IF (YGRD2.EQ.0.) GOTO 998
599	999	WRITE (OUT,45)		612	CALL PLOT1 (YGRD2,RMI1,RMIY)
600		IF (WARN.EQ.46) WRITE (OUT,46)		613	998 IF (YGRD2.LT.0.) IGMT3=1
				614	C --- ABORT
				615	999 IF (WARN.EQ.0.OR.WARN.EQ.55) GOTO 9993
				616	WRITE (OUT,45) WARN
				617	9993 IF (WARN.EQ.46) WRITE (OUT,46)
606		CALL SECH33 (KSCLOS,'WINDAT',6,1,0,RCOD)	5 equal lines	623	REWIND 5
			2 equal lines	624	IF (IGMT3.NE.0) STOP

Table 4.2 (continue)

- Lines 63, 64, 65 and 66 which contained file-manipulating instructions for the PRIME system only and some comments about them were deleted. This was done because these instructions were useless in the IBM 3033 system.

- Two new lines were inserted before line 75 of the old program (line 74 of the new program). The two new lines were inserted there to read the values of the new variables YGRD1, YGRD2 and YGRD. These variables are the Y-dimensions in inches of the plots of undistorted grid, distorted grid and calculated field respectively. They consist a new feature of the program, and thus they do not exist in the old program, which didn't give the plots of undistorted and distorted grid at all, and used fixed Y-dimension for the plot of the calculated electric field in the window.

- Line 159 in the old program was changed to the corresponding line 158 in the new program. Variable K starts from 0 and increases by steps of 10 and this instruction takes care that K do not exceed the number of vertical grid lines (LNVER). But according to the way the instruction was given in the old program, if by chance the total number of vertical lines was less than 10 (very rare case) the program would not pass this check and it would continue to search for non-existing vertical lines.

- Line 169 in the old program was changed to the corresponding line 168 in the new program. The reason for this change is the same as for the previous change, concerning only the number of horizontal grid lines (LNHOR) instead of the number of vertical grid lines (LNVER).

- A line has been added before line 196 of the program. Line 196 gives a message for a very serious error which might occur due to improper grid-data. But, the way the old program was written, the value of the error-monitoring variable WARN wouldn't change and the execution would continue despite the completely wrong data. With the new line inserted, the value of WARN is changed to 1 if the error occurs, and this causes the program to return to MAIN and abort the execution.

- Line 235 in the program has been changed. At this point the program checks if a contour point is within a certain tolerance from any other points of the same contour, and if it is, it follows a certain procedure to move it on that point. The old program would do the same thing for points which coincide, which is usually the case, but in that case this is completely unnecessary and time consuming. With the change which was made, the program simply skips the point if it coincides with an other point.

- Three new lines have been inserted after line 237 of the program. These three lines print a message which

inform the user about points being moved to other points of the contours on which they belong because they were within a certain tolerance from them. This message was not given from the old program.

- A new line has been inserted after line 245 of the old program (or after line 248 of the new program respectively). The program now checks if a point which belongs to a contour is within certain tolerance from points of other contours, and with this instruction it specifies the number of the contour against whose points the current point is checked. Thus, it will be able to inform the user on which contour a point has been moved if such a thing occurs. The old program didn't inform the user at all about this point movement.

- Line 247 of the old program has been changed to the corresponding line 250 of the new program, and

- Three new lines have been inserted after line 249 of the old program (or after corresponding line 253 of the new program). The two changes above are made for exactly the same reason as the two consecutive changes described before the change which is explained previously to the two last ones (ie. they are made for exactly the same reason as the change in line 235 and the insertion after line 237). The only difference is that the program now does not check the contour-points against points of

the same contour, but against points of other contours.

- Five new lines are inserted after line 341 of the old program (or after corresponding line 348 of the new program). This group of five instructions calls subroutine PLOT1 to generate a plot of the undistorted grid. With the first of them, the program resets the value of variable IGMT3 which decides if the execution of the program is to be held after generation of the two grid plots or not. With the second line it examines if the user wants the undistorted grid plot and if not it skips the calling of subroutine PLOT1. Otherwise, it generates the plot of the undistorted grid through PLOT1 and examines if YGRD1 is negative. If it is, it sets IGMT3 equal to 1, so that the execution of the program will be held at the end of subroutine CODING and the user can examine if the grid that he applied is proper.

- Line 446 of the old program is changed to the corresponding line 458 of the new program. Variable ASINB was initialized by the old program here, getting the value of 1000.0 which is supposed to be much bigger than the actual geometrical dimensions of the region in which the electric field is to be calculated. But, if the user uses very small length units to describe a large electrical configuration, which he might do for certain reasons, then they might exceed 1000. , and this results into points

moving to infinity. Variable ASINB must take an initial value much bigger than any geometrical dimension of the problem, and for this reason in the new program it is made to take the value of 100000.

- A new line is inserted after line 499 of the old program, or line 511 of the new program. Here, variable K has a zero value only if the user has given improper grid data and the grid can not be fitted on contour segments, which is a serious error. The old program, after printing the appropriate message, it wouldn't stop but it would start grid fitting again on the same contour segment where the fitting had just failed. It would fail again, start again and so on, thus falling into an infinite loop. With the new instruction inserted here, after the new program prints an error message informing the user about the interruption of line fitting, it branches to label 999 where it exits CODING and is stopped by the MAIN.

- Line 510 of the old program has been changed to corresponding line 523 of the new program. Here, instead of zero, a capital O was punched in the old program. The FORTRAN IV compiler of the PRIME computer system has the feature of setting the values of all variables equal to zero before starting the execution of a program. Thus, the value of variable O was set to zero anyway, and no error occurred from this mispunch. But the IBM 3033 FORTRAN IV compiler

doesn't exhibit this initializing feature, and thus variable O had an arbitrary irregular value, resulting to a generation of an infinite loop or of an error message from the FORTRAN MONITOR program of the IBM 3033.

- Two lines following line 513 of the old program have been deleted, and label 1721 which belonged to the first of them is transferred to the next line (line 516 in the old program, or corresponding line 527 in the new program). The two variables ASSAL and ASCAL which were defined by the two deleted instructions are not used anywhere else in the program, as a thorough scan proved. Consequently, they had no reason to exist there and they were deleted.

- A line is inserted after line 560 of the old program (or after corresponding line 571 of the new program) and

- A line is inserted after line 567 of the old program (or after corresponding line 579 of the new program). Both the above two lines were inserted to make this program compatible with the peculiarities of the Interactive Fortran (IF) Compiler of IBM 3033 computer system. They are not necessary for normal FORTRAN IV compilers and they can be deleted by future users of the program.

- Line 571 of the old program has been changed to corresponding line 584 of the new program. Here, label 184 belongs to a CONTINUE instruction which is followed by the

instruction with label 185, so, instead of branching the program to the CONTINUE instruction and letting it flow to label 185 we branch it right to it.

- Lines 598 to 600 in the old program have been replaced with lines 611 to 617 in the new program. Here, lines 611 to 613 have been actually inserted to control the generation of a plot of the distorted grid in exactly the same way lines 350 to 353 control the generation of the undistorted grid. Line 615 was also placed into the new program because without it the old program generated the message "ERRONEOUS INPUT" to be printed even if the input was correct.

- Line 606 in the old program has been replaced by lines 623 and 624 in the new program. Line 606 in the old program is a file-manipulating instruction of the PRIME computer system which is useless on the IBM 3033. Line 623 in the new program is the one that has replaced it. Line 624 in the new program causes the execution to be held if IGMT3 is not zero, which is the code meaning that the user gave negative YGRD1 and/or YGRD2 in order to stop the program and see if the grid which he applied is proper. This option did not exist in the old program.

- Finally there is one more change in CODING which is not listed in table 3.2 because it was made after this print of *APC was obtained. Line 329 in the old program

which is:

GOTO 402

has been changed to line 336 in the new program which is:

GOTO 412

Here the program searches all the contour points against the line segments which form the contours and, if one point is within a certain tolerance from a line segment, it generates a new point on that segment, thus dividing it into two new segments, and then it moves the contour point on the new point. So, after having moved a point on a segment, the old program was restarting to check all points against all segments from the beginning again thus spending very much time without any reason. After thorough examination, this was decided to be an error, and the new program is branched to label 412 where it takes the next segment instead of taking everything from the beginning again. Personally, I am 99.999% confident that this is correct. The reason that I am not 100% confident is the always conservative Dr. Lauber. Anyway just in case he is right, this change is mentioned particularly, and in case that any malfunction be reported around this area of the problem, the user should simply replace 412 to 402 in line 336 of subroutine CODING of the new program.

4.3 Alterations in EQUIVALENT. The alterations

which were made in subroutine are listed in table 4.3 .

We shall examine them one by one.

- Line 2 in the old edition of the program has been deleted. This was done because line 2 was a special instruction to the PRIME system, which was useless for the IBM 3033 system.

- Line 8 in the old program has been changed to the corresponding line 7 in the new program. As we can see, the dimension of array NTRI(8) was not specified in the old program. It seems that this was accepted by the FORTRAN compiler of the PRIME computer system (although it still intrigues me how could this happen) but it generated a series of compilation-time errors when the program was tested on the IBM 3033 system.

- Line 10 in the old program has been replaced with corresponding lines 9 and 10 in the new program. This was done because variable OUT, which is the logical I/O unit which corresponds to the printer, was declared as INTEGER*2 in the old program. PRIME computer system accepted that, while the IBM 3033 system required that all logical I/O units be specified as INTEGER variables. Thus, variable OUT was taken out of the INTEGER*2 variable group and was specified in a separate INTEGER instruction.

- Six new lines were inserted after line 19 of the program. These six lines initialize the arrays COEFF1 and

```

Unit 0: -CLD lnr 1 len 23 col 1
*****
2      $INSERT $VSUCH>KEYS.F
*****
8          2,TPCS(70),TPCS(100),AUX(500,2),PAUX(500)
*****
10         INTEGER*2 MVERT(70,100),STEP,CUT
*****
After line 19

*****
After line 74
78         NODE(JLAST)=I
*****
308        CALL SECHSS(K$WRIT,'DATA1',5,4,0,NCCD)
309        WRITE(8,821)JFIN
*****
316        823  WRITE(8,821)NODE(J)
*****
320        WRITE(8,825)VOLTS(J)
*****
323        CALL SECHSS(K$CICS,'DATA1',5,4,0,NCCD)
324        CALL SECHSS(K$WRIT,'DATA2',5,5,0,NCCD)
*****
327        WRITE(9,8271)COEFF1(J)
*****
330        CALL SECHSS(K$CICS,'DATA2',5,5,0,NCCD)
*****

```

```

Unit 1: -M88 lnr 1 len 23 col 1
1 equal line *****
After line 1
5 equal lines *****
7          2,TPCS(70),TPCS(100),AUX(500,2),PAUX(500),DIGIT(1
          40),MTRI(8)
1 equal line *****
9          INTEGER*2 MVERT(70,100),STEP
10         INTEGER OUT
9 equal lines *****
20         DO 3 I=1,18145
21           3  COEFF1(I)=0.
22           MINT=0
23           DO 6 I=1,70
24             DO 6 J=1,2
25               6  MDATE1(I,J)=0
55 equal lines *****
81         IF (JF.LE.0) GOTO 741
3 equal lines *****
85         741  NODE(JLAST)=I
229 equal lines *****
315        WRITE(7,821)JFIN
*****
6 equal lines *****
322        823  WRITE(7,821)NODE(J)
3 equal lines *****
326        WRITE(7,825)VOLTS(J)
2 equal lines *****
329        ENDPILR 7
330        REWIND 7
2 equal lines *****
333        WRITE(8,8271)COEFF1(J)
2 equal lines *****
336        ENDPILR 8
337        REWIND 8
2 equal lines *****

```

Table 4.3

NDATLN and also the variable MIRT. The two arrays and the variable were not initialized in the old program, but the program still worked due to the initializing feature of the FORTRAN IV compiler of the PRIME SYSTEM. As we previously said, the IBM 3033 system does not set the values of all variables and arrays equal to zero prior to the execution, and thus the above arrays and variables were not initialized and a series of run-time errors was generated.

- A new line was inserted after line 74 in the old program (or after corresponding line 80 in the new program. This was done because variable JF, which controls a DO-loop, takes the value of zero under certain circumstances. Most probably, the PRIME system accepted the next instruction which is:

```
DO 74 J=1,JF
```

and simply skipped the DO-loop if JF was zero. But the IBM 3033 system was confused if a zero value was assigned to JF and generated a run-time error. Thus, the new instruction was placed there to prevent this from occurring.

- Line 78 in the old program has been changed to the corresponding line 85 in the new program. This was a consequence of the previous change which made the program to skip the DO 74 loop for zero value of JF and branch to the first instruction after it. A label should be assigned

to the instruction where the program is branched, and this is done by this change.

- Line 308 in the old program has been deleted and line 309 has been changed to the corresponding line 315 in the new program. Again here, a special file-manipulating instruction to the PRIME computer system has been deleted and the logical I/O unit 8 in the old program has been changed to logical I/O unit 7 in the new program. Before proceeding any further, we must say that through the whole program some logical I/O units corresponding to storage disk-files have been changed to meet the requirements of the IBM 3033 system. Thus, unit 8 was changed to unit 7 and unit 9 to unit 8, while unit 9 in the new program corresponds to the file where the generated plots are stored. From now on we shall refer to these changes as "logical I/O unit" changes.

- Line 316 in the old program has been changed to corresponding line 322 in the new program, and

- Line 320 in the old program has been changed to corresponding line 326 in the new program. Both the above changes are logical I/O unit changes.

- Lines 323 and 324 in the old program have been changed to corresponding lines 329 and 330 in the new program. Both lines in the old program were special file manipulating instructions to the PRIME system, and were

replaced with equivalent FORTRAN instructions, accepted by the IBM 3033 system.

- Line 327 in the old program has been changed to the corresponding line 333 in the new program. This is one more logical I/O unit change.

- Line 330 in the old program has been changed to the corresponding lines 336 and 337 in the new program. Again here, two FORTRAN file-manipulating instructions accepted by the IBM 3033 system have replaced a special instruction to the PRIME system.

The above are all the alterations which have been made in subroutine EQUIVALENT.

4.4 Alterations in SOLUTN. The alterations which were made in subroutine solutn are listed in table 4.4 . We shall examine them one by one.

Before proceeding any further, we shall define another general category of changes. These are changes concerning special file-manipulating instructions to the PRIME computer system, which are either replaced by equivalent FORTRAN IV instructions accepted by the IBM 3033 computer system or completely deleted. From now on we shall refer to them as "special system-instruction changes".

We shall now continue with the description of alterations in subroutine solutn.

Unit 0: -CID lnr 1 len 24 col 1	Unit 1: -NRW lnr 1 len 24 col 1
*****	*****
2 IINSEFT SYSCCN>KEYS.F	1 equal line
*****	After line 1
6 3,TEMP(7260),JVERT(6,4),NODE4(120),JVERT(2000,3),	3 equal lines
PERM(2000)	5 3,TEMP(7260),JVERT(6,4),NODE4(120),PERM(2000)
7 4,CCEFF(900,9)	6 4,CCEFF(900,9),DTGLEN(140),WTPI(9)
*****	3 equal lines
11 INTEGER*2 MVERT(70,100),CUT	10 INTEGER*2 MVERT(70,100),JVERT(2000,3)
*****	11 INTEGER OUT
74 IF(NZCN(K).EQ.0)GO TO 71	62 equal lines
*****	After line 71
After line 75	1 equal line
*****	75 IF(NZCN(K).EQ.0)GO TO 71
After line 83	8 equal lines
*****	84 IF (ID.EQ.0) GOTO 73
120 2007 PCFMT(1103(10X,15)/3(5X,10.2))	36 equal lines
*****	121 2007 PCFMT(1103(10X,15)/3(5X,10.2))
254 CALL SRCH\$\$ (K\$READ,'DATA1',5,3,0,NCCN)	133 equal lines
*****	After line 254
271 CALL SRCH\$\$ (K\$CIOS,'DATA1',5,3,0,NCCN)	16 equal lines
272 CALL SRCH\$\$ (K\$READ,'DATA2',5,5,0,NCCN)	271 REWIND 7
*****	2 equal lines
275 RPAC(9,3074)TEMP(K)	274 RPAC(9,3074)TEMP(K)
*****	2 equal lines
278 CPIL SRCH\$\$ (K\$CICS,'DATA2',5,5,0,NCCN)	277 REWIND 8
*****	127 equal lines
406 3125 CALL SRCH\$\$ (K\$CICS,'WINDUT',6,2,0,NCCN)	405 3125 ENDPILR 6
*****	406 REWIND 6
*****	2 equal lines
*****	*****

Table 4.4

- Line 2 in the old program has been deleted. This is a special system-instruction change.

- Line 6 and 7 in the old program have been changed to corresponding lines 5 and 6 in the new program. The reason that this change was made is that array IVERT should be specified as an INTEGER*2 array like in all other parts of the program, while in the old program it was left as a simple INTEGER array. Also, the dimensions of arrays DIGLN and NTRI were not specified at all in the old program. This was somehow accepted by the PRIME system FORTRAN IV compiler, while it was rejected by the compiler of the IBM 3033 system, so in the new program the dimensions of the two arrays were specified.

- Line 11 in the old program has been replaced by lines 10 and 11 in the new program. Two changes are involved here in the INTEGER*2 statement. One of them is the addition of array IVERT for reasons which were explained in the description of the previous change. The other change is again associated with variable OUT. As explained before (see description of similar change concerning the same variable in subroutine EQUIVALENT) this variable, which contains the logical I/O unit corresponding to the printer, is accepted to be an INTEGER*2 variable by the PRIME system, but not by the IBM system. Thus, it was specified as a simply INTEGER variable in the new program.

- Line 74 in the old program has been deleted and
- A new line has been inserted after line 75 in the old program (or after corresponding line 74 in the new program). Actually, these two consecutive changes indicate that the sequence between the two following instructions in the old program:

line 74 : IF (NZON(K).EQ.0) GO TO 71

line 75 : ID=NZON(K)

has been inverted in the new program, which now has become

line 74 : ID=NZON(K)

line 75 : IF (NZON(K).EQ.0) GO TO 71

The way the old program was written, if NZON(K) was equal to zero the execution would skip to the instruction labeled 71 before previously storing the value of NZON(K) into variable ID, something which is absolutely necessary. This was considered to be an error in the program and the necessary change was made so that the value of NZON(K) be stored into ID before the skip to label 71.

- A new line has been inserted after line 73 in the program. This change was made because variable ID, which in some cases has the value of zero, specifies a position in an array in the next instruction, and thus it has to be an integer greater than zero. Mysteriously, the PRIME system accepted it to be zero sometimes, probably skipping the following instruction, while IBM 3033 system generated an

error at this point and stopped. Thus, the new line was inserted there to prevent this case and lead the program to skip the next instruction where ID is used as an array index if it has a value of zero.

- Line 120 in the old program has been changed to corresponding line 121 in the new program. As it can be seen, a comma (,) was missing from the FORMAT specification which this line contains, and it was put to its place.

- The rest of the changes in this subroutine are either changes of logical I/O units or special system-instruction changes, which have been explained before and need no further details.

The above are all the alterations which have been made in subroutine Solutn.

4.5 Alterations in PLOTY. The alterations made in subroutine PLOTY are listed in Table 4.5 . As we can see, Ploty is the subroutine in which most alterations were made. This happened because of two reasons: The first reason is that the plot-generation instructions used in the old program belong to the DYNAGRAPHICS package used in the PRIME computer system and they had to be replaced by equivalent plot-generating instructions from the PLOT DESCRIPTION SYSTEM (P.D.S.) which is used by the IBM 3033 computer system. From now on, these changes will be called

Table 4.5

Unit 0: -GID lnr 1 len 23 col 1	Unit 1: -NEW lnr 1 len 23 col 1
***** 3 equal lines *****	***** 3 equal lines *****
4 1,ADPAD(15,15),ADMAD(15,15),DATLN(70,3),TEMP(7260	4 1,ADMAD(15,15),ADMAD(15,15),DATLN(70,3),TEMP(7260
5 1,COEFF(900,9)	5 1,COEFF(900)
5 2,PERM(2000),COEFF1(18145)	5 2,COEFF2(900),COEFF3(900),COEFF4(900),COEFF5(900)
***** 9 equal lines *****	***** 9 equal lines *****
15 EQUIVALENCE (COEFF1(1),TEMP(1)),(COEFF1(7261),CO	17 COMMON/BLOCK33/VGRD
16 1, (COEFF1(15361),PERM(1))	18 EQUIVALENCE (COEFF1(1),TEMP(1)),(COEFF1(7261),CO
***** 13 equal lines *****	***** 13 equal lines *****
30 E5=9.C/E2	37 E5=VGRD/R2
31 IF (E1.GT.E2) E5=10.0/E1	***** 5 equal lines *****
***** 5 equal lines *****	***** 5 equal lines *****
37 CALL GFSEB1	After line 42
38 CALL ENTGRA	***** 5 equal lines *****
***** 5 equal lines *****	***** 5 equal lines *****
44 CALL SLTRZ(0)	48 CALL PALPHA ('ROMAN.3',0)
45 CALL MCVE(-5.0,-5.0)	49 CALL PENCHK ('BLUE')
46 CALL TEXT(IDENT,N)	50 CALL PSYM (1.,1.48,-.28,IDENT,0.,N,0)
47 CALL MCVE(-5.0,-4.7)	51 CALL PHUMBR (1.,1.,.15,PERCV,0.,'PERCENT VOLTS
48 CALL TEXT('PERCENT VOLTS PER LINE:',23)	52 PER LINE: "
49 ENCODE(8,10,IERCV) PERCV	53 1JF5.2*,0)
50 10 FORPAT(PB.3)	54 CALL PHUMBR (-0.0,-0.0,-.15,POTMX,0.,'P. MAX:
51 CALL MCVE(-1.5,-4.7)	55 "JP9.1*,0)
52 CALL TEXT(IPRCV,8)	55 CALL PENCHK ('BLAK')
53 ENCODE(8,10,IERCV) POTMX	***** 5 equal lines *****
54 CALL TEXT(' P. MAX:',7)	***** 5 equal lines *****
55 CALL TEXT(IPRCV,8)	***** 5 equal lines *****
56 ENCODE(8,10,IERCV) ECTYN	***** 5 equal lines *****
57 CALL TEXT(' P. MIN:',7)	***** 5 equal lines *****
58 CALL TEXT(IPRCV,8)	***** 5 equal lines *****
***** 5 equal lines *****	***** 5 equal lines *****
64 E6=-4.5	61 E6=1.
65 COEFF(1,6)=-5.0	62 COEFF(1)=0.
66 COEFF(1,7)=E6	63 COEFF7(1)=E6
67 CCEFF(2,6)=-5.0	64 COEFF6(2)=0.
68 COEFF(2,7)=E5*(E4-E3)+E6	65 COEFF7(2)=E5*(E4-E3)+E6
69 COEFF(3,6)=E5*(E2-E1)-5.0	66 COEFF6(3)=E5*(E2-E1)
70 CCEFF(3,7)=COEFF(2,7)	67 COEFF7(3)=COEFF7(2)
71 COEFF(4,6)=CCEFF(3,6)	68 COEFF6(4)=COEFF6(3)
72 COEFF(4,7)=E6	69 COEFF7(4)=E6
73 CCEFF(5,6)=-5.0	70 COEFF6(5)=0.
74 CCEFF(5,7)=E6	71 COEFF7(5)=E6
75 CALL MCVE(COEFF(1,6),COEFF(1,7))	72 DO 1431 J=1,5
76 DO 1001 JP=2,5	73 COEFF6(J)=COEFF6(J)+1.1
77 1001 CALL CFAM(COEFF(JP,6),COEFF(JP,7))	74 1431 COEFF7(J)=COEFF7(J)+1.1

Table 4.5 (continue).

			75	CALL PLINE (COEP6,COEP7,5,1,0,0,0)
			76	COEP6(1)=COEP6(1)-.1
			77	COEP7(1)=COEP7(1)-.1
			78	COEP6(2)=COEP6(2)-.1
			79	COEP7(2)=COEP7(2)+.1
			80	COEP6(3)=COEP6(3)+.1
			81	COEP7(3)=COEP7(3)+.1
			82	COEP6(4)=COEP6(4)+.1
			83	COEP7(4)=COEP7(4)-.1
			84	COEP6(5)=COEP6(5)-.1
			85	COEP7(5)=COEP7(5)-.1
			86	CALL PLINE (COEP6,COEP7,5,1,0,0,0)
*****		8 equal lines	*****	
86		COEFF(J,6)=E5*(LINC(N1-1)-E1)-5.0	95	COEP6(J)=E5*(LINC(N1-1)-E1)
87	241	COEFF(J,7)=E5*(LINC(N1)-E3)+E6	96	241 COEP7(J)=E5*(LINC(N1)-E3)+E6
88		COEPF(N+1,6)=-5.0	97	COEP6(N+1)=0.
89		COEFF(N+1,7)=-4.5	98	COEP7(N+1)=1.
90		COEPF(N+2,6)=-5.0	99	COEP6(N+2)=0.
91		COEFF(N+2,7)=-4.5	100	COEP7(N+2)=1.
92		CALL MCVE(COEPF(1,6),CCEPF(1,7))	101	DO 2411 J=1,N
93		DO 1002 JP=2,N	102	COEP7(J)=COEP7(J)+1.
94	1002	CALL DEAM(COEPF(JP,6),COEPF(JP,7))	103	2411 COEP6(J)=COEP6(J)+1.
95	242	CONTINUE	104	CALL PLINE (COEP6,COEP7,N,1,0,0,0)
96	C ---	CHANGE COORDINATES	105	DO 2412 J=1,N
97		DC 243 I=1,ITOT	106	COEP7(J)=COEP7(J)-1.
98		COEFF(I,8)=E5*(COEFF(I,8)-E1)-5.0	107	2412 COEP6(J)=COEP6(J)-1.
99	243	COEPF(I,9)=E5*(CCEPF(I,9)-E3)+E6	108	242 CONTINUE
			109	C ---
			110	CALL PENCHG ('RED ')
			111	DO 243 I=1,ITOT
			112	COEP8(I)=E5*(COEP8(I)-E1)
			113	243 COEP9(I)=E5*(COEP9(I)-E3)+E6
*****		29 equal lines	*****	
129		DO 606 I11=8,9	143	IF (I10.EQ.0) GOTO 606
130		K1=I10+I11-7	144	COEP3(J)=COEP8(I11N)+E2*(COEP8(I11N)-COEP8(I11N))
			145	GOTO 6061
131	606	COEFF(J,K1)=COEFF(IMIN,I11)+E2*(COEFF(IMAX,I11)-	146	606 COEP1(J)=COEP8(IMIN)+E2*(COEP8(IMAX)-COEP8(IMIN))
		COEFF(IMIN,I11))	147	6061 IF (I10.EQ.0) GOTO 6062
132		COEFF(J,5)=0	148	COEP4(J)=COEP9(IMIN)+E2*(COEP9(IMAX)-COEP9(IMIN))
			149	GOTO 6063
			150	6062 COEP2(J)=COEP9(IMIN)+E2*(COEP9(IMAX)-COEP9(IMIN))
			151	6063 COEP5(J)=0.
*****		15 equal lines	*****	
148		DC 701 I=1,2	167	COEP6(1)=COEP1(1)
149		CCEFF(1,6)=CCEFF(1,2+I-1)	168	COEP6(2)=COEP1(1)
150	701	COEFF(1,7)=COEFF(1,2+I)	169	COEP7(1)=COEP2(1)
			170	COEP7(2)=COEP4(1)
*****		4 equal lines	*****	
155		IF (CCEFF(1,5).GT.0.5) GO TO 706	175	IF (COEP5(1).GT.0.5) GO TO 706
156		IF (ABS(ABS(COEPF(N,6))-CCEPF(1,1)).LT.0.00001.A	176	IF (ABS(ABS(COEP6(N)-COEP1(1)).LT.0.00001.AND.
	ID.		177	1ABS(COEP7(N)-COEP2(1)).LT.0.00001) GO TO 703
157		1ABS(COEPF(N,7)-CCEPF(1,2)).LT.0.00001) GO TO 703	178	IF (ABS(ABS(COEP6(N)-COEP1(1)).LT.0.00001.AND.
158		IF (ABS(ABS(COEPF(N,6))-CCEPF(1,1)).LT.0.00001.A		
	ID.			

159	1ABS(COEF7(N,7)-COEF7(I,4)).LT.0.00001) GO TO 704		179	1ABS(COEF7(N)-COEF4(I)).LT.0.00001) GO TO 704	
***** 2 equal lines *****			***** 2 equal lines *****		
162	CCEFF(N,6)=CCEFF(I,3)		182	COEP6(N)=COEP3(I)	
163	CCEFF(N,7)=CCEFF(I,4)		183	COEP7(N)=COEP4(I)	
164	CCEFF(I,5)=1		184	COEP5(I)=1	
***** 2 equal lines *****			***** 2 equal lines *****		
167	COEFF(N,6)=COEFF(I,1)		187	COEP6(N)=COEP1(I)	
168	COEFF(N,7)=CCEFF(I,2)		188	COEP7(N)=COEP2(I)	
169	CCEFF(I,5)=1		189	COEP5(I)=1	
***** 12 equal lines *****			***** 12 equal lines *****		
192	IF (ABS(COEFF(I,6))-YEND)**2+(COEFF(I,7)-YEND)**		202	IF (ABS(COEP6(I))-YEND)**2+(COEP7(I)-YEND)**2.GT	
183	2.GT. (ABS(COEFF(N		203	. (ABS(COEP6(N	
184	I,6))-YEND)**2+(COEFF(N,7)-YEND)**2) GO TO 710		204	I))-YEND)**2+(COEP7(N)-YEND)**2) GO TO 710	
185	713 XEND=ABS(COEFF(N,6))		205	713 XEND=ABS(COEP6(N))	
186	YEND=CCEFF(N,7)		206	YEND=COEP7(N)	
187	COEFF(N+1,6)=-5.		207	COEP6(N+1)=0.	
188	COEFF(N+1,7)=-5.		208	COEP7(N+1)=-5	
189	COEFF(N+2,6)=-4.		209	COEP6(N+2)=1.	
190	COEFF(N+2,7)=-4.		210	COEP7(N+2)=1.5	
191	CALL MCVE(COEFF(I,6),COEFF(I,7))		211	DO 7131 J=1,N	
192	DC 1003 JP=2,N		212	COEP7(J)=COEP7(J)+1.1	
	CALL DEAV(COEF(JP,6),COEFF(JP,7))		213	COEP6(J)=COEP6(J)+1.1	
			214	CALL PPLINE (COEP6,COEP7,N,1,0)	
			215	DO 7132 J=1,N	
			216	COEP7(J)=COEP7(J)-1.1	
			217	7132 COEP6(J)=COEP6(J)-1.1	
***** 5 equal lines *****			***** 5 equal lines *****		
198	IF (CCEFF(I,5).GT.0.5) GO TO 711		222	IF (COEP5(I).GT.0.5) GO TO 711	
***** 1 equal line *****			***** 1 equal line *****		
200	DO 710 K=1,5		224	COEP1(N)=COEP1(I)	
201	710 CCEFF(K,K)=COEFF(I,K)		225	COEP2(N)=COEP2(I)	
			226	COEP3(N)=COEP3(I)	
			227	COEP4(N)=COEP4(I)	
			228	COEP5(N)=COEP5(I)	
***** 7 equal lines *****			***** 7 equal lines *****		
209	E1=CCEFF(I,6)		236	E1=COEP6(I)	
210	E2=CCEFF(I,7)		237	E2=COEP7(I)	
***** 1 equal line *****			***** 1 equal line *****		
212	CCEFF(I,6)=COEFF(K,6)		239	COEP6(I)=COEP6(K)	
213	CCEFF(I,7)=COEFF(K,7)		240	COEP7(I)=COEP7(K)	
214	CCEFF(K,6)=E1		241	COEP6(K)=E1	
215	707 CCEFF(K,7)=E2		242	707 COEP7(K)=E2	
***** 3 equal lines *****			***** 3 equal lines *****		
219	CALL EXITGR		246	CALL PLTEND	
***** 2 equal lines *****			***** 2 equal lines *****		

Table 4.5 (continue)

"special graphics-instruction changes". The second reason is that new features were added to this subroutine. These features will be specified when the corresponding changes will be described.

We shall examine the alterations in subroutine PLOTY one by one.

- Lines 4 and 5 in the old program have been changed to the corresponding lines 4 and 7 in the new program. The array COEFF in the old program is a two-dimensional array which contains data for the plot to be generated by the subroutine as well as the final coordinates of the points of lines which are to be plotted finally, obtained after the subroutine has processed the data. The PDS graphics package required that the X- and Y-coordinates of line points be stored into two one-dimensional arrays rather than a two-dimensional one. Thus, the two-dimensional array COEFF(900,9) in the old program was converted to nine one-dimensional arrays COEF1(900), COEF2(900),, COEF9(900) at this point. Each of them corresponds to one of the nine columns of the array COEFF.

- Lines 15 and 16 in the old program have been changed to the corresponding lines 17 to 23 in the new program. As it can be easily seen the change here is that the two-dimensional array COEFF contained in the relevant EQUIVALENCE statement has been divided into the nine

one-dimensional arrays COEF1 to COEF9 because of the reason explained in the description of the previous change. From now on we shall call this change "array division".

- Lines 30 and 31 in the old program have been changed to the corresponding line 37 in the new program. Here, both programs are calculating the scaling factor to be used so that the generated plot has the appropriate dimension. The old program calculated the scale so that the generated plot would have either a 9 inch length or a 10 inch height, depending upon the ratio of these two quantities. This was changed in the new program in such a way that the Y-dimension of the generated plot is the one specified by the user through data-input (variable YGRD).

- Lines 37 and 38 in the old program have been deleted and

- Lines 44 to 58 in the old program have been changed to lines 48 to 55 in the new program. Both these changes are special graphics instructions changes and need no more explanation. These groups of instructions initialize the plot generation procedure and write the title of the plot as well as the percent volts per equipotential line plotted and the maximum and minimum voltage present in the plotted area.

Lines 64 to 77 in the old program have been changed to lines 61 to 86 in the new program. Two changes are made

here: The first is that the old program used the middle of the plot area as origin of the coordinate system ($X=0.$ and $Y=0.$) and placed the plot around this point, while the new program uses the lower left corner of the plot area as origin of the coordinate system and places the plot at the upper right side of this point. The second change is that with instructions 76 through 86 in the new program a second frame is drawn around the orthogonal frame which corresponds to the boundary of the plotted region (window), spaced 0.1" from it. This is a new feature of the program. After this part of the program has been executed, the two orthogonal frames have been drawn.

- Lines 86 to 99 in the old program have been changed to corresponding lines 95 to 111 in the new program. With this group of instructions the program plots the components of the configuration (electrodes, dielectrics etc.) All the changes made here are either the division of the two-dimensional array COEFF into nine one-dimensional arrays COEF1 to COEF9 (lines 86 to 91, 88 and 99 in the old program or corresponding lines 95 to 100, 112 and 113 in the new program), or special graphics instruction changes, or changes such that the origin $X=0.$, $Y=0.$ of the coordinate system is moved from the middle of the plot area, where they were placed in the old program, to the lower left corner of the same area, as the Plot Description System requires.

- Lines 129 to 132 in the old program have been changed to corresponding lines 143 to 151 in the new program. Again here the changes made are related to the division of array COEFF into the nine one-dimensional arrays COEF1 to COEF9.

- Lines 148 to 150 in the old program have been changed the corresponding to lines 167 to 170 in the new program. The reason for this group of changes is the same as for the prior group of changes. The same holds for the next three groups of changes which are:

- Lines 155 to 159 of the old program have been changed to corresponding lines 175 to 179 in the new program,

- Lines 162 to 164 in the old program have been changed to the corresponding lines 182 to 184 in the old program, and

- Lines 167 to 169 in the old program have been changed to the corresponding lines 187 to 189 in the old program.

- Lines 182 to 192 in the old program have been changed to the corresponding lines 202 to 216 in the new program. There were three reasons for the changes made here. The first reason is again the division of array COEFF into nine one-dimensional arrays (lines 182 to 189 of the old program or corresponding lines 202 to 212 in

the new program). The second reason is the change of the origin of the coordinate system from the middle of the plot area (old program) to the lower left corner of the same area (new program). The third reason, finally, is the difference between the special graphics instructions of the PRIME and the IBM 3033 computer system.

The rest of the changes in subroutine PLOTY concern the division of array COEFF into the nine one-dimensional arrays, except the last change which is a special graphics instruction change.

Here we end the description of the alterations made to the program. There are a few more corrections of spelling mistakes in comments placed by Mr. Varela in various positions of the program, but these need no explanation. We shall continue with the description of the new subroutine PLOT1.

5. THE NEW SUBROUTINE "PLOT1"

The new subroutine PLOT1 has been recently added to the program in order to generate plots of either undistorted or distorted grid applied on either the whole region of the configuration or on the window region, depending upon the stage of the program which has been executed when PLOT1 is called.

To find the place where subroutine PLOT1 is called, the reader should refer to section 4.2 "Alterations in CODING" of this report.

The subroutine first specifies a general scaling such that the Y-dimension of the plot that it will generate will be the one specified by the user, except if the corresponding X-dimension then turns out to be bigger than 36 inches. In this case the plot is rescaled so that the X-dimension becomes 36 inches. Then the title, as given by the user, is plotted in blue pencil using the ROMAN.3 set of characters.

The next step is to draw an orthogonal frame which is the boundary of the region to be plotted and another orthogonal frame surrounding it, spaced 0.1" from it. This is done only to make the plot look better, and the black pen is used for the two frames.

After this, the subroutine creates the plots of the components of the configuration (electrodes, dielectrics etc) by joining with straight segments the characteristic points which were given by the user for each contour to describe it. This is also done by the black pen. The way the straight segments which form the contours are generated is the same way used in subroutine PLCTY, as described by Mr. Varela in his report.

Finally, PLOT draws consequent straight segments which form the grid lines, starting first with the vertical grid lines and ending with the horizontal. The red pen is used for this purpose.

This is all the information which regards the new subroutine PLOT1. A detailed flow chart of it is given in figure 5.1 .

Figure 5.1

Figure 5.1 (continue)

6. CONCLUSIONS

As it can be seen from this report, many alterations have been made to the program from the "old" edition which Mr. Varela left in the PRIME system to the "new" edition which is now left in the IBM 3033 computer system at Rensselaer Polytechnic Institute. We tried to correct all the errors of the old program, and we hope that this has been achieved and that the new edition will work well without getting stuck, falling into infinite loops, giving unexplainable messages and generally running into awkward situations for the user. The program was tested with a couple of proper examples. An effort was made that those examples cover most of the cases that the program is supposed to be able to anticipate and it is for sure that the results were very good and were obtained without any trouble at all.

There are still many things that can change in the program until it reaches its final most efficient and economic form. As previously written in this report, it was our purpose to make certain improvements in it, but the number of errors into which we were running into every time we tried to run a new example took all our time on trying to detect them and correct them. Right now, sug-

gested possible improvements are the substitution of the way the main admittance matrix is created and stored, so that both execution time and storage space in computer memory are saved, and the writing of a new subroutine which will make data-input through a digitizer or an interactive CRT graphics terminal by using interactive graphics software. The reason for this last suggestion is that we realize that the present way of data-input is rather difficult and time-consuming. The fact is that this change can not be done once for ever because the program will be used on many different computer systems each of which is going to have its own software and a new subroutine will be required for every different computer system.

Finally, we can say once again that a skilful user with experience in preparing the data for the program is of great importance to its performance and he will increase significantly its efficiency by giving proper grid data, which is the most important part of data-input for the program.