UNLIMITED DISTRIBUTION

1

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

TECHNICAL COMMUNICATION 89/307

July 1989

AD-A212 294

IBM PC ANALYSIS OF POLARIZATION RESISTANCE DATA

A. E. Hardy - C. M. Hanham

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

89 9 13 112

UNLIMITED DISTRIBUTION

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

IBM PC ANALYSIS OF POLARIZATION RESISTANCE DATA

A. E. Hardy - C. M. Hanham

July 1989

Approved by R.S. Hollingshead
H/Dockyard Laboratory Section

Distribution Approved by

[illegible]

D/TD

TECHNICAL COMMUNICATION 89/307

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

ABSTRACT

An IBM PC has been set up to control an EG&G PARC M273 potentiostat to perform electrochemical corrosion experiments using PARC's M342 corrosion measurement software. Modifications to the software permit the measurement of corrosion current over long periods of time using the Polarization Resistance measurement technique. A computer program, PCCORROS, is used to analyze the polarization resistance data to determine the polarization resistance, corrosion current, anodic and cathodic tafel constants, and the corrosion potential.

RÉSUMÉ

Un IBM PC a été configuré de manière qu'il puisse commander un potentiostat EG&G PARC M273 en vue de la réalisation d'expériences sur la corrosion électrochimique au moyen du logiciel de mesure de la corrosion PARC M342. Le logiciel a été modifié de manière à permettre la mesure du courant de corrosion pendant de longs intervalles par la méthode de mesure de la résistance de polarisation. On utilise le logiciel PCCORROS pour analyser les données de résistance de polarisation afin de déterminer la résistance de polarisation, le courant de corrosion, les constantes de Tafel anodique et cathodique, ainsi que le potentiel de corrosion.

TABLE OF CONTENTS

1. INTRODUCTION

The corrosion properties of a metal can be determined by direct methods, such as weight loss measurement and corrosion pit depth measurement, or indirectly by electrochemical techniques. Electrochemical techniques are an important tool in determining the corrosion susceptibility of metals. They utilize a potentiostat or galvanostat to apply a current or voltage to a sample and measure the resulting current or voltage. These methods provide rapid quantitative and qualitative results which can be used to study corrosion under environmental conditions that can be easily controlled in a lab.

DREA Dockyard Laboratory has used two computer controlled corrosion measurement systems from EG&G, Princeton Applied Research Corporation (PARC). The Model 350 (M350) Corrosion Measurement System consists of a central processing unit, a potentiostat and a plotter integrated into a single unit. The Model 351 (M351) is composed of two separate units; a Model 1000 System Processor and a Model 273 (M273) Potentiostat/Galvanostat. It also has a Houston Instrument's HIPLOT DMP-40 plotter for presenting results.

Polarization Resistance measurements are used at DREA to determine the variation of corrosion current with time. They have traditionally been carried out at DREA by using the experiment rerun option on the M350 corrosion measurement system. The data was stored on the DREA DEC20 computer and analyzed using a program called CORROS[1]. It was desirable to perform this same task using the M273 potentiostat in order to carry out simultaneous experiments. Since the M351's software does not provide a rerun option and cannot be modified, it was necessary to purchase a microcomputer to control the M273 potentiostat. An IBM PS/2 Model 50Z personal computer (IBM PC) was obtained along with a National Instruments General Purpose Interface Bus (GPIB) to connect the microcomputer to the M273 potentiostat.

The Model 342 (M342) SOFTCORR Corrosion Measurement Software was obtained from EG&G PARC to be used with the IBM PC to communicate with the M273 potentiostat. The software was written in IBM Advanced BASIC (BASICA) and was modified to perform polarization resistance measurements over long periods of time. CORROS[1] was modified to run on the IBM PC. This modified program is called PCCORROS.

2. IBM PC BASED CORROSION MEASUREMENT SYSTEM

The GPIB was installed in the IBM PC[2,3]. The IBM PC's back-up Reference Disk was updated to include the GPIB in its system configuration. The M342 software was installed on the IBM PC's hard disk. Installation instructions are located on the M342 source disk in the file, README.342.

The M342 software performs the nine standard electrochemical corrosion experiments listed below.

1. Potentiodynamic Polarization
2. Polarization Resistance
3. Potentiokinetic Reactivation
4. Cyclic Polarization
5. Tafel Plot
6. Ecorr vs Time
7. Galvanostatic
8. Potentiostatic
9. Galvanic Corrosion

The M342 software's operating manual[4] explains how to use the software to perform the above experiments.

The M342 software is run by a batch file[5], M342.BAT (Appendix A), which can be executed by typing M342 at the Disk Operating System's[5] (DOS) prompt and pressing the "ENTER" button. A formatted disk, inserted in drive B, is used to store the experimental results and data files (.T and .D files respectively). The DOS command, CHKDSK[5], can be used before an experiment is run to ensure that the disk has sufficient memory to store these files. The memory required depends on the experiment performed and the parameters used. A typical polarization resistance experiment requires 500 bytes of storage space for the results and data files. If a polarization resistance experiment was run once every hour for 200 hours, 100 kbytes of storage space would be required.

Modifications made to S-F.BAS of the M342 software package are listed in Appendix B. They enable experiments to be run over a long period of time by performing subsequent runs of the same experiment. The M342 software limits the experiment name to seven characters[4]. The run number is appended to the experiment name, limiting its length by the number of digits in the run number. For example, run number 123 of experiment FE55 would be saved as experiment FE55123. The data is saved at the end of each run to ensure that it is not lost if the program is inadvertently halted before all the runs are completed.

Before an experiment can be run, the following five questions must be answered.

1. DO YOU WISH TO DO RERUNS (Y=YES; N=NO)?

    The next four questions will not appear and a single experiment will run if you press N and the "ENTER" button.
    To perform an experiment over a long period of time press Y.

2. HOW MANY RUNS DO YOU WANT?

    This is the total number of runs that will be done before the program ends. For example, to do 200 Polarization Resistance experiments on a sample, enter 200.

3. HOW LONG BETWEEN THE START OF EACH RUN (MIN)?

    This is the length of time, in minutes, from the start of one run to the start of the next run. Enter the desired number of minutes and press the "ENTER" button.

4. AT WHAT NUMBER WOULD YOU LIKE TO BEGIN?

    This is the number at which the runs begin to be numbered. If 65 is entered the runs will be numbered 65, 66, 67, ... until the desired number of runs has been performed.

5. CONTINUE (Y=YES)?

    Enter Y to run the experiment. Enter N to change the answers to any of the above questions. The program will start over at question 1.

The computer display between runs is shown in Figure 1. Pressing S advances the program immediately to the next run. Pressing D exits the program to BASICA's DOS shell. All DOS commands, including executing .EXE, .BAT and .COM files, are permitted. BASICA cannot be used from the shell. Typing EXIT and pressing the "ENTER" button exits the shell and returns you to the program. The current directory **MUST** be the PARC subdirectory before exiting the shell or the M342 software will not work properly.

The modifications to the original M342 software can be made, using BASICA, by following the six steps listed below.

The software must be installed, according to the instructions supplied with it, before the modifications can be made.

1. At the DOS prompt type CD\PARC and press "ENTER".
2. Type BASICA and press "ENTER".
3. Type LOAD "S-F.BAS" and press "ENTER".
4. Type in the modifications listed in Appendix B.
5. Type SAVE "S-F.BAS" and press "ENTER".
6. Type SYSTEM and press "ENTER".

The software is now ready to run following the instructions for the original version.

If an experiment that is not part of the M342 software package is needed, it can be created from scratch using HEAD START[6], a software package distributed by EG&G PARC. The M342 software lacks the high quality plots available with the previously used systems. Various plotting packages are available that can take the raw data from any M342 corrosion experiment and make high quality plots. A new, compiled version of the M342 software, which includes an improved plotting utility, is now available from EG&G PARC. Unfortunately the compiled version cannot be modified to perform reruns.

3. PCCORROS

CORROS[1] is written in PASCAL with several subroutines[7] in FORTRAN. It was converted to TurboPASCAL 4.0 in order to run on an IBM PC. The modified version, called PCCORROS, will only run on a computer with a math coprocessor installed. PCCORROS consists of four files, PCCORROS.BAT, DATA.EXE, PCCOR.EXE, and PCCORSUB.TPU.

PCCORROS is run by the batch file PCCORROS.BAT (Appendix C). The data files created by the M342 software are read by the program DATA.EXE (Appendix D), which is written in Microsoft QuickBASIC. It reads the compressed data file from a data disk in drive B and converts it into ASCII (human readable) form. The data, in ASCII form, is stored in the file HOLD.ASC on a disk in drive A. HOLD.ASC is read by PCCOR.EXE (Appendix E), the main program of PCCORROS. PCCOR.EXE performs the analysis of the data until there is "No Further Improvement Available", according to the conditions set up in the original CORROS[1] program, or until 25 iterations have been performed. After 25 iterations, the program will pause and ask the user if he wishes to continue. An answer of 1, for yes, will cause the program to continue running for a maximum of 25 more iterations. Any other answer will cause the program to save the results and terminate the program. PCCORSUB.TPU (Appendix F) is a TurboPASCAL unit used by PCCOR.EXE to perform the tasks of the FORTRAN subroutines of CORROS[1]. Extended real variables[8], with 19 significant digits, were used to ensure the accuracy of the calculations. The results of the analysis are saved in a user specified file with the extension .RES on a disk in drive A. If the file does not exist, a new file is created. Otherwise, the results are appended to the existing file. The raw data is saved on drive A in a file with the same name as the original data file, but with the extension .DAT. Points on the curve that PCCOR.EXE fits to the raw data is stored on drive A in a file with the same name but with the extension .FIT. These files can be printed out to compare the actual data to the fitted curve used by PCCORROS. Figure 2 shows the raw data and the fitted curve on a plot created by ENPLOT[9].

PCCORROS is run by typing PCCORROS at the DOS prompt and pressing the "ENTER" button. An example run of PCCORROS is given in Appendix G. The data analyzed is listed in Appendix H. The results can be printed out by typing in the following line at the DOS prompt.

```
TYPE A:\filename.RES>PRN
```

Where *filename* is the name of the PCCORROS results file to be printed out. An example printout is given in Appendix I.

PCCORROS was used to analyze data from a Tafel plot experiment[10]. Table 1 shows the results of the analysis along with the results from CORROS[1] and the original Tafel plot analysis performed on the same data. The results from PCCORROS and CORROS are both significantly close to the Tafel plot analysis.

A distribution copy of PCCORROS, including its source code, has been made. The program can be installed in an IBM compatible microcomputer that has a math coprocessor. To install PCCORROS, insert the floppy disk into a drive, type *d*: (where *d* is the drive letter) and press the "ENTER" button. Type INSTALL and press "ENTER". The batch file INSTALL.BAT (Appendix J) installs PCCORROS on disk drive C. The files are copied into a subdirectory, PCCORROS, created by INSTALL.BAT. The batch file PCCORROS.BAT is copied to the root directory. PCCORROS is ready to run from the root directory of drive C.

4. CONCLUSION

The system has been set up for ease of use. All of the corrosion experiments and data analyses can be performed from the root directory. The M342 corrosion measurement software has been modified to perform reruns of polarization resistance experiments. This software is difficult to modify due to its complexity and lack of documentation. PCCORROS is easy to run and explains each step to the user as it performs the analyses of polarization resistance data. It can easily be modified, using the source code and the proper compiler, to best suit individual corrosion systems.

Table 1 Results from CORROS[1], PCCORROS, and original Tafel Plot[10] analysis of a Tafel Plot experiment.

	Ecorr (mV)	ba (mV)	bc (mV)	Rp (kohms)	Icorr (mA/cm^2)
Tafel Plot	200	40	120	---	1.0
CORROS	200	40.57±0.71	126.70±13.15	0.01±0.0	1.07±0.03
PCCORROS	200	40.57±0.71	126.70±13.15	0.01±0.0	1.07±0.03

```
           12:59:12
Run 12  will begin in 12 : 16  MIN
Total number of runs: 100
Press 'S' to advance to next run.
Press 'D' to go to DOS shell.  Type EXIT to exit
the DOS shell.  NOTE: You MUST be in the PARC
subdirectory before exiting the DOS shell.
```

Figure 1   Modified M342 corrosion software's display screen between runs.

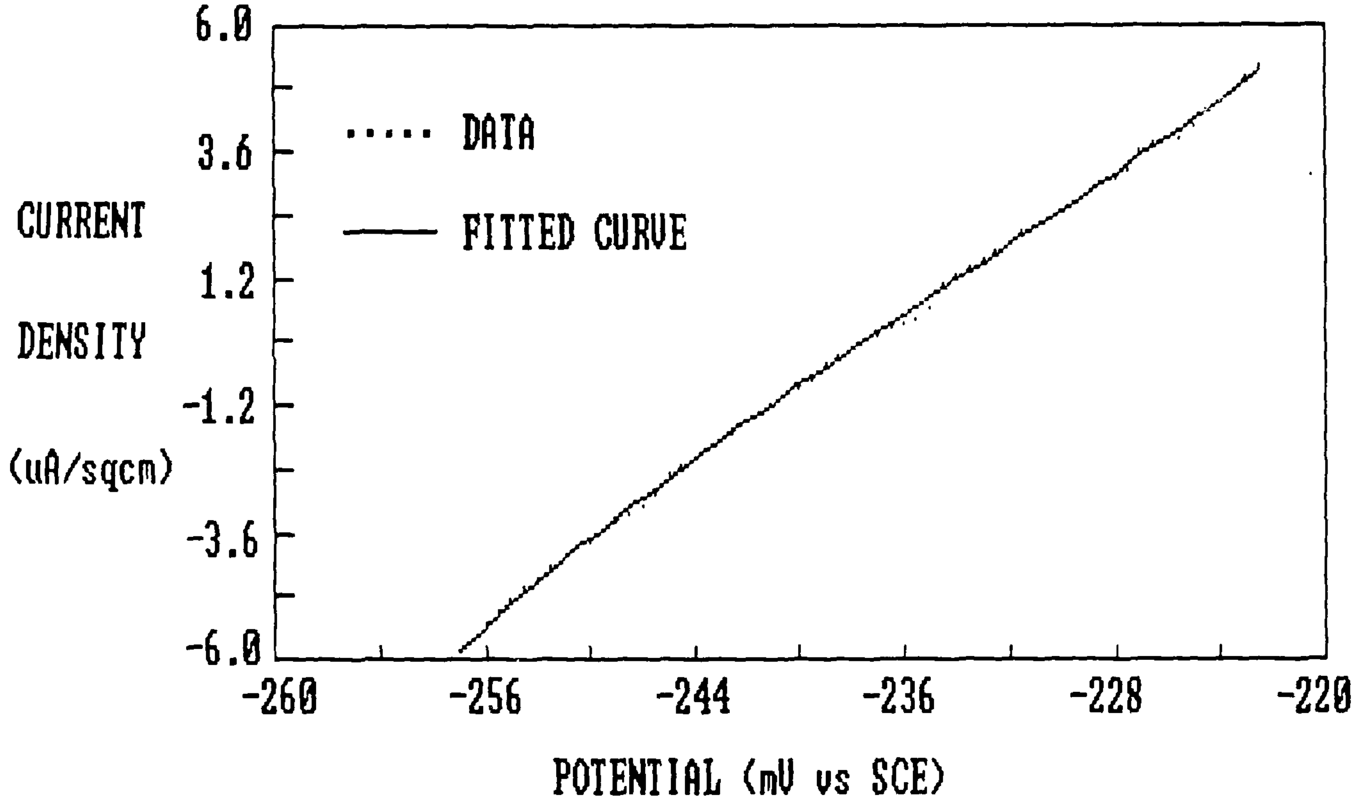

Figure 2 Plot of PCCORROS analysis of data file FE551.DAT.

APPENDIX A

M342.BAT: BATCH FILE THAT RUNS THE M342 SOFTWARE

```
@ECHO OFF
CD\PARC
MODE CO80
BASICA HELLO
CD\
MODE MONO
MENU
```

APPENDIX B

MODIFICATIONS MADE TO S-F.BAS

This allows the M342 corrosion software to perform reruns.

```
15450 CLS:INPUT "               DO YOU WISH TO DO RERUNS (Y=YES;
      N=NO)";RERUN$:IF RERUN$ = "Y" THEN GOSUB 24520  ELSE IF
      RERUN$ <> "N" GOTO 15450
15451 CLS:LOCATE 1,20:PRINT L$;:LOCATE 3,20:PRINT L$:A$=
      E$(EX) + " TECHNIQUE":LOCATE 2:GOSUB 19860
15470 TSTART=TIMER:LOCATE 12:A$="   CHECKING SYSTEM   ":GOSUB
      19860:LOCATE 23:A$=STRING$(40,32):GOSUB 19860:LOCATE
      16:GOSUB 19860:GOTO 17700
17280 R$(0)=E$(EX):A$=R$(1)+".T":DR=2:A=52:GOSUB
     18140:RN=1:GP18140:RN=1:GP=1:GOSUB 15370:IF FRUN% = 1 THEN GOTO
      24700
17281 GOTO 20050

24520 'THIS IS THE RERUN SETUP ROUTINE
24530 INPUT "                     HOW MANY RUNS DO YOU WANT";
      NRUNS%
24535 INPUT "HOW LONG BETWEEN THE START OF EACH RUN (MIN)";
      TRUNS
24540 INPUT "      AT WHAT NUMBER WOULD YOU LIKE TO BEGIN";
      SNRUN%
24545 INPUT "                              CONTINUE (Y=YES)";
      NUE$
24546 IF NUE$ <> "Y" THEN GOTO 15450
24550 TRUNS = TRUNS*60
24560 RCOUNT%=SNRUN% - 1
24570 FRUN%=1
24580 BASE$=R$(1)
24585 GOSUB 24600
24590 RETURN
24600 'THIS IS THE RERUN ROUTINE TO CHANGE THE NAME
24610 RCOUNT%=RCOUNT%+1
24620 ADD$=STR$(RCOUNT%)
24640 ADD$ = RIGHT$(ADD$,(LEN(ADD$)-1))
24650 R$(1) = BASE$ + ADD$
24660 IF RCOUNT%>= (NRUNS%+SNRUN%-1) THEN FRUN%=0
24670 RETURN
24700  'MAIN RERUN ROUTINE
24710 GOSUB 24600
24720 GOSUB 24860
24730 WHILE TIMR < (TSTART+TRUNS) 'LOOP TO WAIT BETWEEN RUNS
24740    GOSUB 24940: WHILE INT(TIMER) = INT(TIMR)
24750    IF INKEY$ = "D" THEN GOSUB 24800 'EXITS TO DOS SHELL
```

```
24760    IF INKEY$ = "S" THEN TIMR=(TSTART+TIMR): WEND
24770 WEND
24780 SCREEN 0,0,0: CLS: WIDTH 80
24790 GOTO 15470
24795 'GO TO DOS SHELL, REINITIALIZE WHEN THE SHELL IS EXITED
24800 CLS: WIDTH 80: SCREEN 0,0,0: SHELL :WIDTH 40
24810 BIB=1146!: BIB728=481!: IBINIT1=65535!-BIB-BIB728-8
24820 IBINIT2=IBINIT1+3: DEF SEG: BLOAD "BIB.M",IBINIT1
24830 IBINIT3=IBINIT1+BIB: BLOAD "BIB728.M",IBINIT3
24840 GOSUB 24860
24850 RETURN
24860 'PRINT OUT DISPLAY BETWEEN RUNS
24865 CLS: SCREEN 0,0,0: PRINT
24870 PRINT "Run     will begin in         MIN":LOCATE 2,4:
      PRINT RCOUNT%
24880 PRINT "Total number of runs:";NRUNS%
24890 PRINT "Press 'S' to advance to next run."
24900 PRINT "Press 'D' to go to DOS shell."
24910 PRINT "Type EXIT to exit the DOS shell.":PRINT
24920 PRINT "NOTE:": PRINT "You MUST be in the PARC
      subdirectory"
24930 PRINT "before exiting the DOS shell."
24940 'PRINTS TIME AND TIME LEFT
24950 LOCATE 1,12: PRINT TIME$: TIMR = TIMER
24960 IF TIMR < TSTART THEN TIMR = TIMR + 86400!
24970 TR%=INT(TRUNS+TSTART-TIMR)+1: TMIN%=TR%\60:
      TSEC%=TR%-TMIN%*60
24980 LOCATE 2,23: PRINT TMIN%;":";TSEC%
24985 RETURN
24990 'LINES CHANGED IN ORDER TO DO RERUNS ARE 15450, 15451,
      15470, 17280, 17281, 24520-24990.
```

APPENDIX C

PCCORROS.BAT: BATCH FILE THAT RUNS PCCORROS

```
ECHO OFF
CLS
CD\PCCORROS
DATA
PCCOR
CD\
```

APPENDIX D

DATA.EXE: READS THE M342 DATA DISKS FOR PCCORROS

```
'This program reads the data saved by the M342 EG&G PARC
'software from Polarization Resistance experiments.  It then
'converts it to ASCII (human readable) form to be analysed by
'the program PCCOR.EXE.  The M342 data disk must be in drive
'A and the PCCORROS results disk must be in drive B.

'February, 1989. Allison Hardy.

'VARIABLES:
'CURRENT%:  RAW DATA IN THE FORM SAVED BY BASICA
'R$:        SETUP PARAMETERS IN THE .T FILE CREATED BY THE
   M342 SOFTWARE
'FNVALR:    CONVERTS A R$ VALUE INTO NUMERICAL FORM TO USE IN
   CALCULATIONS
'FNA1,FNA2,FNCALC:  USED TO CONVERT THE CURRENT DATA INTO
   uA/CM^2
'B1,B2:     CONSTANTS
'NAMEOFILE$:   M342 POLARIZATION RESISTANCE DATA FILE
'DFILE$:       NAMEOFILE$ WITH THE PATH AND DRIVE ADDED
'OUTFILE$:     OUTPUT FILE, HOLD.ASC, ON DRIVE A: USED BY
   PCCOR.EXE
'NOFPOINTS%:   NUMBER OF DATA POINTS
'STYLE$:       OUTPUT FORMAT FOR THE FILE HOLD.ASC
'EREF:         REFERENCE POTENTIAL (Ecorr OR 0)
'VOLTINIT:     INITIAL POTENTIAL
'VOLTSEND:     FINAL POTENTIAL
'DVOLTS:       INCREMENTAL POTENTIAL
'VOLTS:        POTENTIAL IN mV VERSES REFERENCE ELECTRODE
'AMPS:         CURRENT DENSITY IN uA/CM^2


DIM CURRENT%(4100)
DIM R$(53)
ON ERROR GOTO ERRORHANDLER
'CALCULATIONS
B1 = 4096
B2 = 2048
DEF FNVALR (X) = VAL(R$(X))
DEF FNA1 (X) = INT(X / B1)
DEF FNA2 (X) = X - FNA1(X) * B1
DEF FNCALC (X) = -(FNA2(X) + B1 * (FNA2(X) > B2)) * (10 ^
                (FNA1(X) + 3)) / (FNVALR(27) - 1 *
                (FNVALR(27) = 0))
CLS
```

```
READDATA:
PRINT "INSERT M342 DATA DISK INTO DRIVE B:."
PRINT "INSERT PCCORROS RESULTS DISK IN DRIVE A:."
INPUT "Data File: "; NAMEOFILE$
DFILE$ = "B:\" + NAMEOFILE$
OUTFILE$ = "A:\HOLD.ASC"
OPEN DFILE$ + ".T" FOR INPUT AS #1
FOR I = 0 TO 52
   INPUT #1, R$(I)
NEXT I
CLOSE #1
IF R$(0) <> "POLN RESISTANCE" THEN
   PRINT "**This file is not a M342 PARC Polarization
            Resistance data file.**"
   INPUT "Do you wish to continue with the analysis?";ANSWER$
   IF ANSWER$ <> "Y" THEN
      GOTO READDATA
   END IF
END IF
NOFPOINTS% = FNVALR(13)
BLOAD DFILE$ + ".D", VARPTR(CURRENT%(0))
OUTPUTDATA:
STYLE$ = " #######.####   #######.####"
OPEN OUTFILE$ FOR OUTPUT AS #2
PRINT #2, NAMEOFILE$
FOR I = 1 TO NOFPOINTS%
   IF RIGHT$(R$(2), 1) = "E" THEN EREF = FNVALR(20) ELSE EREF
   = 0
   VOLTSINIT = FNVALR(2) + EREF
   VOLTSEND = FNVALR(3) + EREF
   DVOLTS = FNVALR(21) * SGN(VOLTSEND - VOLTSINIT)
   VOLTS = VOLTSINIT + (I - 1) * DVOLTS
   AMPS = FNCALC(CURRENT%(I))
   PRINT #2, USING STYLE$; AMPS, VOLTS
NEXT I
CLOSE #2

END

ERRORHANDLER:
        IF ERR = 53 THEN     'FILE NOT FOUND
                PRINT "File not found!"
                RESUME READDATA
        ELSE
                PRINT "Error "; ERR; " occured."
                ON ERROR GOTO 0
        END IF
```

APPENDIX E

PCCOR.EXE: MAIN PROGRAM USED BY PCCORROS

```
PROGRAM CorrosionAnalysis;{$N+}

{A program to find the corrosion potential Ecorr, the anodic}
{and cathodic tafel constants ba and bc, the polarization   }
{resistance Rp and the corrosion current density Icorr, from}
{the mixed potential region around Ecorr.  The data is read }
{from a polarization resistance data file created by the    }
{EG&G PARC Model 342 Corrosion Measurement Software using   }
{another program, DATA.EXE.  The program's basic routine is }
{a nonlinear least squares fit of the data. Output includes }
{the corrosion parameters, their estimated errors, and the  }
{relative RMS error of the fitted data.  These parameters   }
{are stored in a user named output file on drive A. This    }
{program was originally written for use on the DEC-20 and   }
{has been modified for use on the IBM PS/2 M50Z using Turbo }
{Pascal 4.0.          }

USES PCCORSUB;
{Contains procedures spline, decomp, solve, and seval.}


VAR
   DATA,PLOTS,LIST,OUT                                 :text;
   gradstore,Jac                                       :matrix;
   gradient,mgradient,work                             :svector;
   potential,current,voltdata,currdata,BB,CC,DD :vector;
   ipvt                                                :isvector;
   i,j,m,n,numofpts,count,ans,asn                      :integer;
   Rp,Ecorr,Icorr,ba,bc,cond                           :real;
   detailed                                            :boolean;
   results,prin,infile                                 :string;


PROCEDURE dataheader;
CONST
   size  = 6;
   blank = ' ';
TYPE
   wordtype = ARRAY[1..size] of char;

VAR
   ch                 :char;
   i                  :integer;
   seive,keyword      :wordtype;
```

```
PROCEDURE skipblanks;
BEGIN
   REPEAT read(DATA,ch) UNTIL (ch<>blank);
END;

PROCEDURE initseive;
VAR     i :integer;
BEGIN
   skipblanks;
   seive[1] :=ch;
   FOR i := 2 TO size DO
   BEGIN
      read(DATA,ch);
      seive[i] := ch;
   END;
END;

PROCEDURE lookfor(keyword:wordtype);
VAR i      : integer;
BEGIN
   initseive;
   REPEAT
      write(seive[1]);
      read(DATA,ch);
      FOR i := 1 TO size-1 DO
         seive[i] :=  seive[i+1];
      seive[size] := ch;
   UNTIL (seive = keyword);
   writeln;
   FOR i := 1 TO size DO
      write(seive[i]);
END;

BEGIN {dataheader}
   writeln('Would you like to read the introduction? yes =
1');
   read(ans);
   IF ans = 1  THEN
   BEGIN
      writeln('PCCORROS is a program to find the corrosion
potential Ecorr, the anodic');
      writeln('and cathodic tafel constants ba and bc, the
polarization resistance Rp,');
      writeln('and the corrosion current density Icorr, from
the mixed potential region');
      writeln('around Ecorr. The data is read from a
polarization resistance data file,');
      writeln('created by the EG&G PARC Model 342 Corrosion
Measurement System, by');
```

```
      writeln('another program, DATA.EXE. The program can be
run in two modes: detailed');
      writeln('or simple. The only difference is the amount
of output to the screen.');
      writeln('In detailed mode the program outputs the
gradient, the Jacobian and its');
      writeln('construction, the correction vector and its
treatment at each iteration.');
      writeln('The simple mode operates the same routines but
suppresses output.');

   END;
   writeln;
   writeln('Do you want a detailed run? yes = 1');
   readln;read(ans);
   IF ans = 1 THEN detailed := TRUE
      ELSE detailed := FALSE;
   repeat
      writeln;
      assign(DATA,'a:\HOLD.ASC');
      {$I-}
      reset(DATA);
      {$I+}
      I := IOResult;
      IF I <> 0 THEN
         begin
         writeln('FILE DOES NOT EXIST: MAKE SURE RESULTS DISK
IS IN DRIVE A');
         writeln('PRESS "ENTER" TO CONTINUE');
         readln;
         readln;
         end;
      writeln;
   UNTIL (I = 0);
   readln(DATA,infile);
END;{dataheader}


PROCEDURE DataGenerator;

{Takes raw data from the data file HOLD.ASC on drive A and}
{converts current density from uAmps per sqcm to mAmps per}
{sqcm and passes a cubic spline through the data to allow }
{access to points 'between' data points.                  }

VAR
   voltage,amperage,vstp : real;

PROCEDURE SCALE;  {Scales uAmps to mAmps}
BEGIN
```

```
   FOR i := 1 TO numofpts DO
   BEGIN
      current[i] := current[i] /1.0E3;
   END;
END;

PROCEDURE reorder; {Reorders decreasing voltage data}
VAR    tempvolt,tempamp   :real;
       i,nexti            :integer;
BEGIN
   writeln('reordering');
   FOR i := 1 TO (numofpts DIV 2) DO
   BEGIN
      nexti             :=numofpts + 1-i;
      tempvolt          :=potential[i];
      tempamp           :=current[i];
      potential[i]      :=potential[nexti];
      current[i]        :=current[nexti];
      potential[nexti]  :=tempvolt;
      current[nexti]    :=tempamp;
   END;
END;{reorder}

PROCEDURE viewdata;
BEGIN
   writeln;
   writeln('Do you want to see the data tabulated ? yes =
1');
   readln;read(ans);
   IF ans = 1 THEN
   BEGIN

writeln('*****************************************************
******');
      writeln;
      writeln('   Potential       Current       Potential
Current');
      writeln('    (mVolts)       (mAmps)         (mVolts)
(mAmps)');
      writeln;
      FOR j := 1 TO numofpts DIV 2 DO
         writeln(j:2,potential[j]:8:2,current[j]:15:6,
                (j+numofpts DIV 2):8,potential[j+numofpts DIV
                 2]:8:2, current[j+numofpts DIV 2]:16:6);
writeln('*****************************************************
****');
   END;
END;{viewdata}

BEGIN {datagenerator}
```

```
   i := 0;
   WHILE NOT eof(DATA) DO
   BEGIN
      i := i + 1;
      read(DATA,current[i],potential[i]);
   END;{WHILE}
   numofpts := i-1;
   scale;
   writeln(numofpts:4,' data points read in');
   IF potential[1] > potential[4] THEN reorder;
   viewdata;
   spline(numofpts,potential,current,BB,CC,DD);
END; {datagenerator}

PROCEDURE screendata(window : real);
{Screens out data more than a 'window' away from Ecorr.}

BEGIN
   FOR i := 1 TO numdata DO
   BEGIN
      voltdata[i] :=0;  currdata[i] :=0;
   END;
   j := 0;
   FOR i := 1 TO numofpts DO
   BEGIN
      IF abs(potential[i] - Ecorr) <= window  THEN
      BEGIN
         j := j+1;
         voltdata[j] := potential[i];
         currdata[j] :=current[i];
      END;
   END;
   count := j;
   writeln(count:4,'points in range');
END; {screendata}


PROCEDURE datafit;
{Let L=ln(10), Ecorr be the corrosion potential and E-Ecorr}
{be the overpotential, oP, in millivolts.  Let I be the    }
{total current density in mAmps/cm^2 and Icorr be the      }
{corrosion current density in mAmps/cm^2. Then the equation}
{that governs I wrt P is:                                  }
{    1.    I=Icorr*(exp(L*oP/ba)-exp(-L*oP/bc))            }
{where ba and bc are the anodic and cathodic Tafel         }
{constants.  Further, the slope of the I vs oP curve at    }
{oP=0 is a constant @ oP=0, (dI/doP)=1/Rp, where Rp is the }
{polarization resistance and @ oP=0,  (dI/doP)=Icorr*ln(10)}
{*(1/ba+1/bc).  Thus                                       }
{    2.    1/Icorr=L*Rp*(1/ba+1/bc).                       }
```

```
{Now substitute u=L/ba and v=L/bc and 2. into 1. and get:  }
{    3.    I=(exp(oP*u)-exp(-oP*v)))/(Rp*(u+v)).            }
{Now sustitute u=w+a and v=w-a and get:                     }
{    4.    I=exp(P*a)*sinh(P*w)/(Rp*w)                      }
{The variables in equation 4 can be partly or wholly        }
{separated.  Consider I=I(oP): i)I(oP)/I(-oP)=-exp(2*oP*a) }
{contains only a, ii)sqrt(I(oP)*-I(-oP))=sinh(oP*w)/(Rp*w) }
{only Rp and w, or substituting Icorr into (ii),            }
{iii) sqrt(I(oP)*-I(-oP))=2*Icorr*sinh(oP*w).  One variable}
{which has not been delt with yet, Ecorr, is hidden in P or}
{oP and since these appear as arguments to the exponential }
{it is very necessary to accurately determine Ecorr.  The  }
{initial approximation to Ecorr is the potential E at which}
{the current density I(E)=0.  This is found by solving for }
{the root of spln(E)=0, where spln is the cubic spline     }
{which interpolates the data and this is done using        }
{Newton's method, x1=x0-y/y'.                               }

VAR
   V,oP,oPincr,minoP,w0,sum1,sum2,sum3,sum4,yval,dyval,
   temp1,temp2,L,kp,delta,w,sumw,sdevw,estEcorr,range,a,
   suma,sdeva,den,z,IoP,Iplus,Iminus,Ip3half,Im3half,
   oPrange,rms,newrms,epsilon,alpha,beta,gamma,sdevIcorr,
   sdevba,sdevbc,sdevRp,X : real;
   i,j,num,rej,iterations : integer;

FUNCTION sinh(X:real):real;
VAR Xext   :extended;
BEGIN
   Xext := X;
   Xext := (exp(Xext)-exp(-Xext))/2;
   sinh := Xext;
END;

FUNCTION eval(V:real):real;
{Returns from the data spline a current value for a given}
{input potential in mvolts.}
BEGIN
   seval(numofpts,V,potential,current,BB,CC,DD,yval,dyval);
   eval := yval;
END;

FUNCTION deval(V:real) :real;
{Returns the rate of change of current at a potential.}
BEGIN
   seval(numofpts,V,potential,current,BB,CC,DD,yval,dyval);
   deval := dyval;
END;

FUNCTION fitcurrent(V,Ecorr,ba,bc,Icorr:real):real;
```

```
BEGIN
   oP := V-Ecorr;
   IF oP <> 0   THEN
      fitcurrent := Icorr*(exp(L*oP/ba)-exp(-L*oP/bc))
   ELSE
      fitcurrent := 0.0;
END;

FUNCTION rmserr(Ecorr,ba,bc,Icorr:real):real;
VAR      sum1,sum2    :real;
         i,j          :integer;
BEGIN
   sum1 := 0;sum2 := 0;j := 0;
   FOR i := 1 TO count DO
   BEGIN
      oP := voltdata[i] - Ecorr;
      IF ((abs(oP)>=minoP)   AND (abs(oP)<=oPrange+1))   THEN
      BEGIN
         sum1 := sum1 + sqr(currdata[i] - fitcurrent(oP
+Ecorr,Ecorr,ba,bc,Icorr));
         sum2 := sum2 + sqr(currdata[i]);
         j := j+1;
      END;
   END;
   rmserr := sqrt(sum1/sum2);
END;

PROCEDURE improve(VAR Ecorr,ba,bc,Icorr :real);
{This routine takes initial values of Ecorr, ba, bc, and   }
{Icorr and returns improved values, least squares sense, by}
{applying a non-linear Newton iteration in four variables. }
{The iteration is of the form J0*dX=-G(X) where dX is the  }
{correction vector, J0 is the Jacobian matrix evaluated at }
{X0 and G0 is the gradient of the sum of squared errors at }
{X0, the initial parameters.  Since there are orders of    }
{magnitude difference between the actual parameters Ecorr, }
{ba, bc and Icorr this routine accepts these values as     }
{constants and creates its own set of variables epsilon,   }
{alpha, beta, and gamma. These new variables are put into  }
{the equation for I as multipliers of Ecorr, ba, bc, and   }
{Icorr respectively.  They are chosen this way so that each}
{of their initial values will be 1(one), negating any size }
{effect that would have existed if Ecorr, ba, bc, and Icorr}
{were used directly.  After each successful iteration the  }
{'constants' Ecorr, ba, bc and Icorr are corrected by      }
{multiplication by their corresponding new variable and the}
{new variables are reset to 1(one) for the next iteration. }
{The procedure is continued until epsilon, alpha, beta and }
{gamma remain constant at a value of 1(one) or the routine }
{iterates past its maximum allowed runs (user set, default }
{of 50).                                                   }

VAR
temp1,temp2,predcurr,error,
pepsilon,palpha,pbeta,pgamma                       :real;
i,j,m,n                                            :integer;

PROCEDURE grad(eps,alp,beta,gam :real);
VAR    maxgrad : real;
       i        :integer;
BEGIN
     {'p' is used to represent differentiation}
   pepsilon := 0;pgamma := 0;palpha := 0;pbeta := 0;
   FOR i := 1 TO count DO
   BEGIN
      oP := voltdata[i] - eps*Ecorr;
      IF ((abs(oP) >= minoP)  AND (abs(oP)<=oPrange+1)) THEN
      BEGIN
         predcurr := fitcurrent(voltdata[i], eps*Ecorr,
alp*ba, beta*bc, gam*Icorr);
         error := currdata[i] - predcurr;
         temp1 := exp(L*oP/ba)/ba;
         temp2 := exp(-L*oP/bc)/bc;
         pepsilon := pepsilon + (2*error*Icorr*L*Ecorr*
(temp1+temp2));
         pgamma := pgamma +(-2*error*predcurr);
         palpha := palpha + (2*error*Icorr*L*oP*temp1);
         pbeta := pbeta + (2*error*Icorr*L*oP*temp2);
      END;
   END;
   gradient[1]:=pepsilon;
   gradient[2]:=palpha;
   gradient[3]:=pbeta;
   gradient[4]:=pgamma;
END; {grad}


PROCEDURE jacobian(Ecorr,ba,bc,Icorr :real);
{The Jacobian matrix is made up of the derivatives of}
{the gradient in the form:Jac[i,j] = dgrad[i]/dX[i]. }
{This routine approximates the derivatives by a      }
{divided difference and further assures that the     }
{matrix is exactly symmetric, accounting for some    }
{roundoff.                                           }

VAR    m,n,i    :integer;
CONST del = 1E-5;
BEGIN
   grad(1+del,1,1,1);
   FOR m := 1 TO 4 DO
```

```
      gradstore[1,m]:=gradient[m];
   grad(1,1+del,1,1);
   FOR m := 1 TO 4 DO
      gradstore[2,m] := gradient[m];
   grad(1,1,1+del,1);
   FOR m := 1 TO 4 DO
      gradstore[3,m] := gradient[m];
   grad(1,1,1,1+del);
   FOR m := 1 TO 4 DO
      gradstore[4,m] := gradient[m];
{Writes out the gradient at the 4 points around current}
{estimate, for use in finding Jacobian.}
   IF detailed THEN
   BEGIN
      writeln;
      writeln('gradient at each parameter + del');
      FOR m := 1 TO 4 DO
      BEGIN
         FOR n := 1 TO 4 DO
         write(gradstore[m,n]:18);
         writeln;
      END;
      writeln;
   END;
   grad(1,1,1,1);
   IF detailed THEN
   BEGIN
      writeln('gradient: dEcorr,dba,dbc,dIcorr');
      FOR m:=1 TO 4 DO
         write(gradient[m]:18);
      writeln;writeln;
   END;
   FOR m:= 1 TO 4 DO
      FOR n := 1 TO 4 DO
         Jac[m,n] := (gradstore[n,m]-gradient[m])/del;

         {Writes out the Jacobian before forced symmetry.}
   IF detailed THEN
   BEGIN
      writeln('Jacobian before symmetry check');
      FOR m := 1 TO 4 DO
      BEGIN
         FOR n := 1 TO 4 DO
            write(Jac[m,n]:18);
         writeln;
      END;
   END;
   FOR m:= 1 TO 4 DO
      FOR n:= 1 TO 4 DO
      BEGIN
```

```
         IF m<>n   THEN
         IF m<n    THEN Jac[m,n] := (Jac[m,n]+Jac[n,m])/2;
      END;
   FOR m := 1 TO 4 DO
      FOR n := 1 TO 4 DO
         IF m>n   THEN Jac[m,n]:=Jac[n,m];
   IF detailed THEN
   BEGIN
      writeln;
      writeln('Jacobian matrix');
      FOR m := 1 TO 4 DO
      BEGIN
         FOR n := 1 TO 4 DO
            write(Jac[m,n]:18);
         writeln;
      END;
      writeln;
   END;
   FOR m := 1 TO 4 DO
   mgradient[m] := -gradient[m];
END; {jacobian}


BEGIN {improve}
   jacobian(Ecorr,ba,bc,Icorr);
   decomp(order,order,Jac,cond,ipvt,work);
   IF abs(cond - 1) < 1E-6  THEN
   BEGIN
      IF detailed THEN
         writeln('Jacobian is singular to working
precision.');
      FOR n := 1 TO 4 DO
         mgradient[n] := 20* mgradient[n];
      IF detailed THEN
         writeln('Negative gradient followed to a new
point.');
   END
   ELSE
   BEGIN
      solve(order,order,Jac,mgradient,ipvt);
   END;
   IF detailed  THEN
   BEGIN
      writeln('Initial correction vector.');
      FOR n := 1 TO 4 DO
         write(mgradient[n]:18);
      writeln;
   END;
   WHILE((abs(mgradient[1])>0.5)
      OR(abs(mgradient[2])>0.5)
```

```
        OR (abs(mgradient[3])>0.5)
        OR (abs(mgradient[4])>0.5))DO
     BEGIN
        FOR i := 1 TO 4 DO
        mgradient[i] := mgradient[i]/2;
        IF detailed THEN
           write(' 50% parameter change exeeded:');
        IF detailed THEN
           writeln(' Reduced correction by half');
     END; {while}
     newrms := rmserr(Ecorr*(1+mgradient[1]), ba*(1+
mgradient[2]), bc*(1+mgradient[3]),Icorr*(1+mgradient[4]));
     j := 0;
     WHILE ((newrms > rms) AND (j < 25)) DO
     BEGIN
        FOR i := 1 TO 4 DO
           mgradient[i] := mgradient[i]/2;
        IF detailed THEN
           writeln('Error overshoot: reduced correction by
half');
        newrms := rmserr(Ecorr*(1+mgradient[1]),ba*(1+
mgradient[2]), bc*(1+mgradient[3]),Icorr*(1+mgradient[4]));
        j := j+1;
        IF j = 25 THEN
        begin
           writeln('No further improvement available.');
           ans := 99;
        end;
     END;{while}
     epsilon   :=1+mgradient[1];
     alpha     :=1+mgradient[2];
     beta      :=1+mgradient[3];
     gamma     :=1+mgradient[4];
     IF ((abs(mgradient[1])<1E-6) AND (abs(mgradient[2])<1E-6)
AND (abs(mgradient[3])<1E-6) AND (abs(mgradient[4])<1E-6))
THEN
     BEGIN
        writeln('No further improvement available.');
        ans := 99;
     END;
if j = 25 THEN
     newrms := rms;
ELSE BEGIN
     Ecorr :=Ecorr*epsilon;
     ba    :=ba*alpha;
     bc    :=bc*beta;
     Icorr :=Icorr*gamma;
END;
     IF((ba <= 0)OR(bc <= 0))  THEN
     BEGIN
```

```
      writeln('Negative tafel constants reached.');
   END;

   Rp := ((ba*bc)/(ba+bc))/(L*Icorr);
   if detailed then begin
      writeln;
      writeln(' Ecorr       ba        bc        Icorr Rp');
      writeln(Ecorr:8:2,ba:10:2,bc:10:2,Icorr:12:6,Rp:12:0);
   end;
END;{improve}

PROCEDURE stats;
VAR i     :integer;
BEGIN
   j:=0; m:=0 ; n:=0;
   sdevIcorr :=0; sdevba:=0; sdevbc:=0; sdevRp:=0;
   FOR i := 1 TO count DO
   BEGIN
      oP := voltdata[i]-Ecorr;
      IF ((abs(oP)>=minoP) AND (abs(oP)<=oPrange))  THEN
      BEGIN
         j         :=j+1;
         IoP       :=currdata[i];
         temp1     :=exp(L*oP/ba);
         temp2     :=exp(-L*oP/bc);
         sdevIcorr :=sdevIcorr+sqr(Icorr-IoP/(temp1-temp2));
         sdevRp    :=sdevRp+sqr(Rp-((temp1- temp2)/(L*(1/ba +
1/bc) *IoP)));
         IF oP > 0 THEN
         BEGIN
            m:=m+1;
            sdevba:=sdevba+sqr(ba-L*oP/ln(temp2+IoP/Icorr))
         END
         ELSE
         BEGIN
            n:=n+1;
            sdevbc·=sdevbc+sqr(bc+L*oP/ln(temp1-IoP/Icorr));
         END;
      END;
   END;
   sdevIcorr :=sqrt(sdevIcorr/(j-1));
   sdevRp    :=sqrt(sdevRp/(j-1));
   sdevba    :=sqrt(sdevba/(m-1));
   sdevbc    :=sqrt(sdevbc/(n-1));
END;{stats}
```

```
BEGIN {datafit}
   L:=ln(10);
{This is Newton's iteration loop for Ecorr.  The initial}
{guess is the rest potential Erest.}
   estEcorr := (potential[1]+potential[numofpts])/2;
   REPEAT
      Ecorr := estEcorr;
      estEcorr :=  Ecorr-(eval(Ecorr)/deval(Ecorr));
   UNTIL(abs((estEcorr-Ecorr)/Ecorr)<=1E-6);
   Ecorr := estEcorr;
   writeln('Estimated Ecorr is: ',Ecorr:8:2);
   IF abs(potential[1] - Ecorr) <=
abs(potential[numofpts]-Ecorr)
     THEN oPrange := abs(potential[1]-Ecorr) + 0.000001
   ELSE oPrange :=abs(potential[numofpts]-Ecorr) + 0.000001;
   writeln('Range around Ecorr reduced to: ',oPrange:4:1);
   range := oPrange;
   screendata(range);
   sum1:=0; sum2:=0; sum3:=0;
   oP:=5;i:=0;
   REPEAT
      i:=i+1;
      Iplus := eval(oP+Ecorr);
      Iminus := eval(-oP+Ecorr);
      a := ln(abs(Iplus/Iminus))/(2*oP);
      IF odd(i)   THEN
      BEGIN
         write('  at oP=',oP:5:1);
         write('   a = ',a:8:6);
      END
      ELSE
      BEGIN
         write('   at oP=',oP:5:1);
         writeln('   a = ',a:8:6);
      END;
      sum1 := sum1 + a;
      oP := oP + 0.5;
   UNTIL (oP > oPrange);
   num := i;
   a := sum1/num;
   writeln('Average a is: ',a:8:6);
   REPEAT
      write('Enter starting oP: ');
      writeln('minimum oP where a becomes reasonably
constant');
      readln;read(minoP);ans := 99;
      IF minoP <= 2  THEN
      BEGIN
         writeln('WARNING : do not start at less than 2');
         ans := 98;
```

```
          END
          ELSE IF minoP >=oPrange  THEN
          BEGIN
             writeln('WARNING : data only good to ',oPrange:4:1);
             ans := 98;
          END;
       UNTIL(ans <> 98);
       i:=0;sum1:=0;sum2:=0;
       oP:=minoP;
       REPEAT
          Iplus    :=eval(oP+Ecorr);
          Ip3half  :=eval(1.5*oP+Ecorr);
          Iminus   :=eval(-oP+Ecorr);
          Im3half  :=eval(-1.5*oP+Ecorr);
          temp1  :=sqrt(abs((Ip3half*Im3half)/(Iplus*Iminus)))/2;
          temp1  :=(temp1+sqrt(sqr(temp1)+1))/2;
          IF temp1 > 1.0   THEN
          BEGIN
             w:=2*ln(temp1+sqrt(sqr(temp1)-1))/oP;
             i:=i+1;
             sum1:=sum1+w;
          END
          ELSE
             writeln('Data too scattered at ',oP:4:1,' to give a
 w.');
          oP:=oP+0.5;
       UNTIL(oP>=2*oPrange/3);
       num:=i;
       w:=sum1/num;
       writeln('Average w is: ',w:8:6);
       i    :=0;
       sum1 :=0;
       oP   :=minoP;
       REPEAT
          Iplus :=eval(oP+Ecorr);
          Iminus:=eval(-oP+Ecorr);
          a     :=ln(abs(Iplus/Iminus))/(2*oP);
          sum2  :=sum2+a;
          Rp    :=sinh(w*oP)/(w*(sqrt(abs(Iplus*Iminus))));
          sum1  :=sum1+Rp;
          i     :=i+1;
          oP    :=oP+0.5;
       UNTIL(oP >= oPrange);
       num  :=i;
       a    :=sum2/num;
       Rp   :=sum1/num;
       writeln('Re-estimate of a is: ',a:6:4,' Average Rp is:
 ',Rp:8:0);
       Icorr:=1/(Rp*2*w);
       writeln('Estimated Icorr is: ',Icorr:10:8);
```

```
sum1:=0;i:=0;oP:=-oPrange;
REPEAT
   IF (abs(oP)>=minoP) THEN
   BEGIN
      i     :=i+1;
      IoP   :=abs(eval(oP+Ecorr));
      temp1 :=IoP/(2*Icorr*exp(a*oP));
      w     :=(ln(temp1+sqrt(sqr(temp1)+1)))/(abs(oP));
      IF detailed THEN
         writeln(' at ',oP:4:1,' w = ',w:8:6);
      sum1  :=sum1+w;
   END;
   oP := oP +0.5;
UNTIL (oP>=oPrange);
num  :=i;
w    :=sum1/num;
writeln('Re-estimate of w is: ',w:10:8);
ba   :=L/(w+a); bc:=L/(w-a); Icorr:=1/(2*w*Rp);
writeln('Initial Ecorr,ba,bc,Icorr and Rp are:');
writeln(Ecorr:6:2,ba:10:2,bc:10:2,Icorr:12:6,Rp:12:0);
rms  :=rmserr(Ecorr,ba,bc,Icorr);
writeln('Initial relative RMS error is: ',rms:8:6);
writeln('Enter 1 if acceptable.');
readln;read(ans);
WHILE ans <> 1 DO
BEGIN
   writeln('Enter ba bc Rp  *in mvolts/decade and ohms*');
   readln;read(ba,bc,Rp);
   Icorr :=((ba*bc)/(ba+bc))/(L*Rp);
   rms   :=rmserr(Ecorr,ba,bc,Icorr);
   writeln('Relative RMS error = ',rms:8:6);
   writeln('Enter 1 if acceptable.');
   readln;read(ans);
END;
iterations := 0;
ans := 1;
REPEAT
   improve(Ecorr,ba,bc,Icorr);
   iterations := iterations + 1;
   rms := newrms;
   if iterations = 25 then
      begin
      iterations := 0;
      writeln('25 iterations have been done.  Do you wish
      to do 25 more? (1=YES)'); readln;read(ans);
      if ans <> 1 then
         ans := 99;
      end;
if detailed then
      writeln('Relative RMS error now = ',rms:8:6);
   UNTIL ans=99;
   stats;
   writeln('Do you want results on the printer ?(Y = yes)');
   readln;read(prin);
   if prin='Y' THEN
      ASSIGN(OUT,'LPT1')
   ELSE
      ASSIGN(OUT,'CON');
   append(OUT);
   writeln;
   writeln(OUT);
   writeln(OUT,'---------------------------------------');
   writeln(OUT,'File ',infile);
   writeln(OUT,'relative RMS error = ',rms:8:6);
   writeln(OUT,'Ecorr = ',Ecorr:7:2,' millivolts');
   writeln(OUT,'Icorr = ',Icorr*1E3:7:5,'
   +/-',sdevIcorr*1E3:7:5,' microamps/cm^2'); writeln(OUT,'ba
   = ',ba:6:2,' +/- ',sdevba:6:2,' millivolts');
   writeln(OUT,'bc    = ',bc:6:2,' +/- ',sdevbc:6:2,'
millivolts');
   writeln(OUT,'Rp    = ',Rp*1E-3:6:2,' +/-
',sdevRp*1E-3:5:2,' kohms');
   writeln(OUT,'---------------------------------------');
   writeln(OUT);
   CLOSE(OUT);
   repeat
      writeln('Results FILE name ?');
      readln;read(results);
   until results <> '';
   assign(LIST,'a:\'+results+'.res');
   {$I-}
   append(LIST);
   {$I+}
   if IOResult <> 0 then
      rewrite(LIST);
   writeln(LIST);
   writeln(LIST,'---------------------------------------');
   writeln(LIST,'File ',infile);
   writeln(LIST,'relative RMS error = ',rms:8:6);
   writeln(LIST,'Ecorr = ',Ecorr:7:2,' millivolts');
   writeln(LIST,'Icorr = ',Icorr*1E3:7:5,'
   +/-',sdevIcorr*1E3:7:5,' microamps/cm^2');
   writeln(LIST,'ba = ',ba:6:2,' +/- ',sdevba:6:2,'
   millivolts');
   writeln(LIST,'bc    = ',bc:6:2,' +/- ',sdevbc:6:2,'
millivolts');
   writeln(LIST,'Rp    = ',Rp*1E-3:6:2,' +/-
',sdevRp*1E-3:5:2,' kohms');
   writeln(LIST,'---------------------------------------');
   writeln(LIST);
   close(LIST);

   ASSIGN(PLOTS,'A:\'+INFILE+'.DAT');
   ASSIGN(LIST,'A:\'+INFILE+'.FIT');
   REWRITE(PLOTS);
   REWRITE(LIST);
   FOR I := 1 TO  COUNT DO
      BEGIN
      WRITELN(PLOTS,VOLTDATA[I]:15,', ',CURRDATA[I]*1E3:15);
      WRITELN(LIST,VOLTDATA[I]:15,', ',FITCURRENT
(VOLTDATA[I], Ecorr,ba,bc,Icorr)*1E3:15);
      END;
   CLOSE(PLOTS);
   CLOSE(LIST);
END;{datafit}


BEGIN {main}

   dataheader;
{Opens the data file and reads in the M342 data file name. }
{Displays the instructions and sets up a simplified or     }
{detailed run.                                             }

   datagenerator;
{Reads in raw data in mVolts and uAmps and stores it in    }
{arrays POTENTIAL and CURRENT in mVolts and mAmps. POTENTIAL}
{is checked to ensure it is in ascending order, then an    }
{interpolating cubic spline is passed through the data.    }

   datafit;
{Determines constants Ecorr, ba, bc, Icorr and Rp by first }
{making substitutions for the constants then separating the}
{equation into parts, each with one or two constants.  Then}
{solves each equation for approximations to its constant(s).}
{These are then used as the initial parameters to a non-   }
{linear least squares fitting routine which calculates the }
{gradient and the Jacobian matrix to the sum of squared    }
{errors and generates a correction to each parameter.  The }
{iteration continues until the size of the corrections     }
{become small enough that no further improvement can be    }
{made.  The constants are then output with ther estimated  }
{error and the predicted data are tabulated with the actual}
{data.                                                     }

END.{main}
```

APPENDIX F

PCCORSUB.TPU: TURBOPASCAL UNIT USED BY PCCOR.EXE

Contains subroutines SOLVE[7], DECOMP[7], SEVAL[7] and SPLINE[7].

```
UNIT PCCORSUB;{$N+}
INTERFACE
CONST  numdata = 200;
       order = 4;
TYPE   index = 1..numdata;
       sindex = 1..order;
       svector = ARRAY[sindex] of real;
       isvector = ARRAY[sindex] of integer;
       vector = ARRAY[index] of real;
       matrix = ARRAY[sindex,sindex] of real;

PROCEDURE SPLINE(N:integer;X,Y:vector;VAR B,C,D:vector);
PROCEDURE DECOMP(NDIM,N:integer;VAR A:matrix;VAR COND:real;
          VAR IPVT:isvector;VAR WORK:svector);
PROCEDURE SEVAL(N:integer;U:real;X,Y,B,C,D:vector;VAR VALU,
          DVAL:real);
PROCEDURE SOLVE(NDIM,N:integer;VAR A:matrix;VAR B:svector;VAR
          IPVT: isvector);

IMPLEMENTATION
PROCEDURE SOLVE;
{Solution of linear system, A*X=B.  Not used if DECOMP}
{has detected singularity.                            }
{INPUT:                                               }
{NDIM= Declared row dimension of array containing A.  }
{N   = Order of matrix.                               }
{A   = Triangularized matrix obtained from DECOMP.    }
{B   = Right hand side vector.                        }
{IPVT= Pivot vector obtained from DECOMP.             }
{OUTPUT:                   }
{B   = Solution vector, X.  }

VAR
   KB,KM1,NM1,KP1,I,K,M :INTEGER;
   T                    :real;

BEGIN
IF (N <> 1) THEN {Forward elimination}
BEGIN
   NM1 := N - 1;
   FOR K := 1 TO NM1 DO
   BEGIN
        KP1 := K + 1;
```

```
          M := IPVT[K];
          T := B[M];
          B[M] := B[K];
          B[K] := T;
          FOR I := KP1 TO N DO
                  B[I] := B[I] + A[I, K] * T;
     END;
  {Back substitution}
     FOR KB := 1 TO NM1 DO
     BEGIN
          KM1 := N - KB;
          K := KM1 + 1;
          B[K] := B[K] / A[K, K];
          T := -B[K];
        FOR I := 1 TO KM1 DO
                  B[I] := B[I] + A[I, K] * T;
     END;
  END;
  B[1] := B[1] / A[1, 1];
  END;

  PROCEDURE DECOMP;
  {Decomposes a real matrix by Gaussian elimination and        }
  {estimates the condition of the matrix.                     }
  {Uses SOLVE to compute solutions to linear systems.          }
  {                                                            }
  {INPUT:                                                      }
  {NDIM = Declared row dimension of the array containing A.    }
  {N    = Order of the matrix.                                 }
  {A    = Matrix to be triangularized.                         }
  {                                                            }
  {OUTPUT:                                                     }
  {A contains an upper triangular matrix U and a permutated    }
  {version of a lower triangular matrix I-L so that:           }
  {(permutation matrix)*A = L*U.                               }
  {                                                            }
  {COND = An estimate of the condition of A.                   }
  {For the linear system A*X=B, changes in A and B may cause   }
  {changes COND times as large in X.  If COND+1.0 = COND, A    }
  {is singular to working precision.  COND=1.0+32 if exact     }
  {singularity is detected.                                    }
  {                                                            }
  {IPVT = The pivot vector.                                    }
  {IPVT(K) = The index of the K-th pivot row.                  }
  {IPVT(N) = (-1)^(number of interchanges)                     }
  {                                                            }
  {Work space:  The vector work must be declared and included}
  {in the call.  Its input contents are ignored.  Its output   }
  {contents are usually unimportant.                           }
  {                                                            }
```

```
{The determinant of A can be obtained on output by          }
{DET(A) = IPVT(N)*A(1,1)*A(2,2)*...*A(N,N).                 }

VAR
    NM1, I, J, K, KP1, KB, KM1, M : INTEGER;
    EK, ANORM, YNORM, ZNORM, T    :EXTENDED;
    AT : ARRAY[1..4,1..4] OF EXTENDED;
    CONDT :EXTENDED;
    WORKT :ARRAY[1..4] OF EXTENDED;

PROCEDURE CHANGES;
VAR       I,J:INTEGER;
{CHANGE EXTENDED VARIABLES TO REAL}
BEGIN
   COND := CONDT;
   FOR I := 1 TO N DO
   BEGIN
      FOR J := 1 TO N DO
         A[I,J] := AT[I,J];
      WORK[I] := WORKT[I];
   END;
END;
   {END OF CHANGES}

BEGIN {CHANGE REAL VARIABLES TO EXTENDED}
   CONDT := COND;
   FOR I := 1 TO N DO
   BEGIN
      FOR J := 1 TO N DO
         AT[I,J] := A[I,J];
      WORKT[I] := WORK[I];
   END; {END OF CHANGES}
   IPVT[N]  := 1;
   IF (N <> 1) THEN
   BEGIN {Compute 1-norm of A}
     NM1 := N - 1;
     ANORM := 0.0;
     FOR J := 1 TO N DO
     BEGIN
        T := 0;
        FOR I := 1 TO N DO
                T := T + ABS(AT[I, J]);
        IF (T > ANORM) THEN ANORM := T;
     END;
     {Gaussian elimination with partial pivoting}
     FOR K := 1 TO NM1 DO
     BEGIN
        KP1 := K + 1;
        M := K;
        {Find pivot}
```

```
          FOR I := KP1 TO N DO
          BEGIN
               IF (ABS(AT[I, K]) > ABS(AT[M, K])) THEN M := I;
          END;
          IPVT[K] := M;
          IF (M <> K) THEN IPVT[N] := -IPVT[N];
          T := AT[M, K];
          AT[M, K] := AT[K, K];
          AT[K, K] := T;
          {Skip step if pivot is zero.}
          IF (ABS(T) > 1E-9) THEN
          BEGIN
               {Compute multipliers.}
               FOR I := KP1 TO N DO
                  AT[I, K] := -AT[I, K] / T;
               {Interchange and eliminate by columns.}
               FOR J := KP1 TO N DO
               BEGIN
                  T := AT[M, J];
                  AT[M, J] := AT[K, J];
                  AT[K, J] := T;
                  IF (ABS(T) > 1E-9) THEN
                  BEGIN
                      FOR I := KP1 TO N DO
                          AT[I, J] := AT[I, J] + AT[I, K] * T;
                  END;
               END;
            END;
         END;
{COND = (1-NORM of A)*(an estimate of 1-NORM of A-inverse) }
{Estimate obtained by one step of inverse iteration for the}
{small singular vector.  This involves solving two systems }
{of equations, (A-transpose)*Y=E and A*Z=Y where E is a    }
{vector of +1 or -1 chosen to cause growth in Y.           }
{Estimate = (1-NORM of Z)/(1-NORM of Y)                    }
{Solve(A-TRANSPOSE)*Y=E                                    }

     FOR K := 1 TO N DO
     BEGIN
         T := 0.0;
         IF (K > 1) THEN
         BEGIN
            KM1 := K - 1;
            FOR I := 1 TO KM1 DO
               T := T + AT[I, K] * WORKT[I];
         END;
         EK := 1;
         IF (T < 0) THEN EK := -1.0;
         IF (ABS(AT[K, K]) < 1E-9) THEN
         BEGIN
```

```
                CONDT :=1.0+32;
                CHANGES;
                EXIT;
             END;
             WORKT[K] := -(EK+T) / AT[K,K];
         END;
         FOR KB := 1 TO NM1 DO
         BEGIN
             K := N - KB;
             T := 0;
             KP1 := K + 1;
             FOR I := KP1 TO N DO
                 T := T + AT[I, K] * WORKT[I];
             WORKT[K] := T;
             M := IPVT[K];
             IF (M <> K) THEN
             BEGIN
                T := WORKT[M];
                WORKT[M] := WORKT[K];
                WORKT[K] := T;
             END;
         END;
         YNORM := 0;
         FOR I := 1 TO N DO
         BEGIN
            YNORM := YNORM + ABS(WORKT[I]);
         END;
         {Solve A*Z=Y}
         CHANGES;
         SOLVE(ndim,n,A,work,ipvt);
         FOR I := 1 TO N DO
         BEGIN
            FOR J := 1 TO N DO
               AT[I,J] := A[I,J];
            WORKT[I] := WORK[I];
         END;
         ZNORM := 0;
         FOR I := 1 TO N DO
         BEGIN
           ZNORM := ZNORM + ABS(WORKT[I]);
         END;
         {Estimate condition}
         CONDT := ANORM * ZNORM / YNORM;
         IF (CONDT < 1.0) THEN  CONDT := 1.0;
         CHANGES;
         EXIT;
    END;
    {1-BY-1}
    CONDT := 1;
    IF (ABS(AT[1, 1]) > 1E-9) THEN
```

```
BEGIN
   CHANGES;
   EXIT;
END;
{Exact singularity.}
CONDT := 1.0+32;
CHANGES;
END;

PROCEDURE SEVAL;
{This subroutine evaluates the cubic spline function and  }
{its derivative and returns VAL and DVAL.                 }
{VAL  = Y(I)+B(I)*(U-X(I))+C(I)*(U-X(I))^2+D(I)*(U-X(I))^3}
{DVAL = B(I)+2*C(I)*(U-X(I))+3*D(I)*(U-X(I))^2            }
{Where X(I) < U < X(I+1), using Horner's rule.            }
{If U < X(1) then I=1 is used.                            }
{If U >= X(N) then I=N is used.                           }
{                                                         }
{INPUT:                                                   }
{N   = The number of data points.                         }
{U   = The abscissa at which the spline is to be evaluated.}
{X,Y = The arrays of data abscissas and ordinates.         }
{B,C,D = Arrays of spline coefficients computed by spline. }
{                                                          }
{If U is not in the same interval as the previous call,    }
{then a binary search is performed to determine the proper }
{interval.                                                 }

VAR
    I, J, K : INTEGER;
    DX : real;

BEGIN
   I :=1;
   IF (U >= X[I]) THEN
      IF (U <= X[I+1]) THEN
      BEGIN
         DX := U-X[I];
         VALU := Y[I]+DX*(B[I]+DX*(C[I]+DX*D[I]));
         DVAL := B[I]+DX*(2*C[I]+DX*(3*D[I]));
         EXIT;
      END;
   I := 1;
   J := N+1;
   {Binary search.}
   REPEAT
      K:=(I+J) DIV 2;
      IF (U < X[K]) THEN J:=K;
      IF (U >= X[K]) THEN I:=K;
   UNTIL (J <= (I+1));
   {Evaluate spline and derivative.}
   DX := U - X[I];
   VALU := Y[I] + DX * (B[I] + DX * (C[I] + DX * D[I]));
   DVAL := B[I] + DX * (2 * C[I] + DX * (3 * D[I]));
END;

PROCEDURE SPLINE;
{The coefficients B(I), C(I), and D(I), I=1,2,...,N are    }
{computed for a cubic interpolating spline.                 }
{S(X) = Y(I)+B(I)*(X-X(I))+C(I)*(X-X(I))^2+D(I)*(X-X(I))^3}
{for X(I) <= X <= X(I+1)                                     }
{                                                          }
{INPUT:                                                    }
{N = The number of data points or knots (N >= 2)           }
{X = The abscissas of knots in strictly increasing order}
{Y = The ordinates of the knots                          }
{                                                        }
{OUTPUT:                                                 }
{B,C,D = Arrays of spline coefficients as defined above}
{Using P to denote differentiation,                      }
{Y(I) = S(X(I))                                          }
{B(I) = SP(X(I))                                         }
{C(I) = SPP(X(I))/2                                      }
{D(I) = SPPP(X(I))/6 (Derivative from the right.)        }
{The procedure SEVAL is used to evaluate the spline and}
{its derivative.                                         }

VAR
    NM1, IB, I : INTEGER;
    T          :real;
BEGIN
   NM1 := N - 1;
   IF (N < 2) THEN EXIT;
   IF (N >= 3) THEN
   BEGIN
      {Set up tridiagonal system.}
      {B=Diagonal, D=Offdiagonal, C=Right hand side}
      D[1] := X[2] - X[1];
      C[2] := (Y[2] - Y[1]) / D[1];
      FOR I := 2 TO NM1 DO
         BEGIN
            D[I] := X[I + 1] - X[I];
            B[I] := 2 * (D[I - 1] + D[I]);
            C[I + 1] := (Y[I + 1] - Y[I]) / D[I];
            C[I] := C[I + 1] - C[I];
         END;
      {End conditions.  Third derivative at X(1) and X(N)}
      {obtained from divided differences.                }
      B[1] := -D[1];
      B[N] := -D[N - 1];
      C[1] := 0;
      C[N] := 0;
      IF (N <> 3) THEN
      BEGIN
         C[1] := C[3] / (X[4] - X[2]) - C[2] / (X[3] - X[1]);
         C[N] := C[N - 1] / (X[N] - X[N - 2]) - C[N - 2] /
(X[N - 1] - X[N - 3]);
         C[1] := C[1] * SQR(D[1]) / (X[4] - X[1]);
         C[N] := -C[N] * SQR(D[N - 1]) / (X[N] - X[N - 3]);
      END;
      {Forward elimination}
      FOR I := 2 TO N DO
      BEGIN
        T := D[I - 1] / B[I - 1];
        B[I] := B[I] - T * D[I - 1];
        C[I] := C[I] - T * C[I - 1];
      END;
      {Back substitution}
      C[N]:= C[N] / B[N];
      FOR IB := 1 TO NM1 DO
      BEGIN
        I := N - IB;
        C[I] := (C[I] - D[I] * C[I + 1]) / B[I];
      END;
      {Compute polynomial coefficients.}
      B[N] := (Y[N] - Y[NM1]) / D[NM1] + D[NM1] * (C[NM1] + 2
* C[N]);
      FOR I := 1 TO NM1 DO
      BEGIN
           B[I] := (Y[I + 1] - Y[I]) / D[I] - D[I] * (C[I+1]
+ 2 * C[I]);
           D[I] := (C[I + 1] - C[I]) / D[I];
           C[I] := 3 * C[I];
      END;
      C[N] := 3 * C[N];
      D[N] := D[N - 1];
      EXIT;
   END;
   B[1] := (Y[2] - Y[1]) / (X[2] - X[1]);
   C[1] := 0;
   D[1] := 0
END;
END.
```

APPENDIX G

SIMPLIFIED PCCORROS RUN

```
Would you like to read the introduction? yes = 1
2

Do you want a detailed run? yes = 1
2

 100 data points read in

Do you want to see the data tabulated ? yes = 1
2
Estimated Ecorr is:  -237.69
Range around Ecorr reduced to: 15.3
  62points in range
  at oP=  5.0   a = -0.008587   at oP=  5.5   a = -0.005321
  at oP=  6.0   a = -0.002811   at oP=  6.5   a = -0.002065
  at oP=  7.0   a = -0.003319   at oP=  7.5   a = -0.006285
  at oP=  8.0   a = -0.008199   at oP=  8.5   a = -0.006307
  at oP=  9.0   a = -0.006264   at oP=  9.5   a = -0.006371
  at oP= 10.0   a = -0.007122   at oP= 10.5   a = -0.004228
  at oP= 11.0   a = -0.002421   at oP= 11.5   a = -0.003676
  at oP= 12.0   a = -0.005147   at oP= 12.5   a = -0.005323
  at oP= 13.0   a = -0.003477   at oP= 13.5   a = -0.003513
  at oP= 14.0   a = -0.003755   at oP= 14.5   a = -0.003211
  at oP= 15.0   a = -0.003721Average a is: -0.004815
Enter starting oP: minimum oP where a becomes reasonably
constant
5
Data too scattered at  5.0 to give a w.
Data too scattered at  8.0 to give a w.
Data too scattered at  8.5 to give a w.
Data too scattered at  9.0 to give a w.
Average w is: 0.054312
Re-estimate of a is: -0.0048 Average Rp is:     3072
Estimated Icorr is: 0.00299712
Re-estimate of w is: 0.05433495
Initial Ecorr,ba,bc,Icorr and Rp are:
-237.69     46.50     38.93    0.002996       3072
Initial relative RMS error is: 0.026588
Enter 1 if acceptable.
1

 No further improvement available.
Do you want results on the printer ?(Y = yes)
N
```

```
---------------------------------------
File FE551
relative RMS error = 0.020926
Ecorr = -237.56 millivolts
Icorr = 3.69377 +/- 0.08267 microamps/cm^2
ba    =  55.47  +/-  1.81 millivolts
bc    =  47.97  +/-  1.40 millivolts
Rp    =   3.02  +/- 0.07 kohms
---------------------------------------

Results FILE name ?
TEST
```

APPENDIX H

DATA FILE FE551.DAT

```
POTENTIAL(mV)     CURRENT(mA/cm^2)

-2.530000E+0002,  -5.991200E+0000
-2.525000E+0002,  -5.591800E+0000
-2.520000E+0002,  -5.297500E+0000
-2.515000E+0002,  -5.066200E+0000
-2.510000E+0002,  -4.814000E+0000
-2.505000E+0002,  -4.624800E+0000
-2.500000E+0002,  -4.456600E+0000
-2.495000E+0002,  -4.267400E+0000
-2.490000E+0002,  -4.057200E+0000
-2.485000E+0002,  -3.868000E+0000
-2.480000E+0002,  -3.741900E+0000
-2.475000E+0002,  -3.552700E+0000
-2.470000E+0002,  -3.384500E+0000
-2.465000E+0002,  -3.258400E+0000
-2.460000E+0002,  -3.048100E+0000
-2.455000E+0002,  -2.863100E+0000
-2.450000E+0002,  -2.495300E+0000
-2.445000E+0002,  -2.308200E+0000
-2.440000E+0002,  -2.144200E+0000
-2.435000E+0002,  -2.009700E+0000
-2.430000E+0002,  -1.866700E+0000
-2.425000E+0002,  -1.713300E+0000
-2.420000E+0002,  -1.503000E+0000
-2.415000E+0002,  -1.311800E+0000
-2.410000E+0002,  -1.156200E+0000
-2.405000E+0002,  -9.838000E-0001
-2.400000E+0002,  -8.829000E-0001
-2.395000E+0002,  -7.105000E-0001
-2.390000E+0002,  -4.436000E-0001
-2.385000E+0002,  -2.733000E-0001
-2.380000E+0002,  -9.520000E-0002
-2.375000E+0002,   4.370000E-0002
-2.370000E+0002,   1.308000E-0001
-2.365000E+0002,   2.674000E-0001
-2.360000E+0002,   3.597000E-0001
-2.355000E+0002,   4.303000E-0001
-2.350000E+0002,   6.243000E-0001
-2.345000E+0002,   1.082600E+0000
-2.340000E+0002,   1.259200E+0000
-2.335000E+0002,   1.400000E+0000
-2.330000E+0002,   1.530400E+0000
```

```
-2.325000E+0002,  1.698500E+0000
-2.320000E+0002,  1.881400E+0000
-2.315000E+0002,  2.053800E+0000
-2.310000E+0002,  2.194700E+0000
-2.305000E+0002,  2.297700E+0000
-2.300000E+0002,  2.465800E+0000
-2.295000E+0002,  2.667600E+0000
-2.290000E+0002,  2.882100E+0000
-2.285000E+0002,  2.993500E+0000
-2.280000E+0002,  3.079700E+0000
-2.275000E+0002,  3.233100E+0000
-2.270000E+0002,  3.607300E+0000
-2.265000E+0002,  3.760800E+0000
-2.260000E+0002,  3.819600E+0000
-2.255000E+0002,  3.861700E+0000
-2.250000E+0002,  4.061400E+0000
-2.245000E+0002,  4.393500E+0000
-2.240000E+0002,  4.498600E+0000
-2.235000E+0002,  4.750900E+0000
-2.230000E+0002,  5.003200E+0000
-2.225000E+0002,  5.213400E+0000
```

APPENDIX I

PRINTOUT OF A PCCORROS RESULTS FILE

```
---------------------------------------
File FE551
relative RMS error = 0.020926
Ecorr = -237.56 millivolts
Icorr = 3.69377 +/- 0.08267 microamps/cm^2
ba    =  55.47 +/-   1.81 millivolts
bc    =  47.97 +/-   1.40 millivolts
Rp    =   3.02 +/-  0.07 kohms
---------------------------------------


---------------------------------------
File FE552
relative RMS error = 0.030113
Ecorr = -244.75 millivolts
Icorr = 1.74461 +/- 0.06352 microamps/cm^2
ba    =  37.89 +/-   1.20 millivolts
bc    =  40.86 +/-   2.23 millivolts
Rp    =   4.89 +/-  0.17 kohms
---------------------------------------


---------------------------------------
File FE553
relative RMS error = 0.087937
Ecorr = -238.68 millivolts
Icorr = 1.99252 +/- 0.11613 microamps/cm^2
ba    =  33.01 +/-   1.67 millivolts
bc    =  54.15 +/-   3.65 millivolts
Rp    =   4.47 +/-  0.26 kohms
---------------------------------------


---------------------------------------
File FE554
relative RMS error = 0.034270
Ecorr = -235.64 millivolts
Icorr = 1.79764 +/- 0.08619 microamps/cm^2
ba    =  33.65 +/-   1.24 millivolts
bc    =  37.62 +/-   2.59 millivolts
Rp    =   4.29 +/-  0.19 kohms
---------------------------------------
```

```
---------------------------------------
File FE555
relative RMS error = 0.027763
Ecorr = -233.08 millivolts
Icorr = 1.84006 +/- 0.06822 microamps/cm^2
ba    =  33.49 +/-   1.29 millivolts
bc    =  36.24 +/-   1.69 millivolts
Rp    =   4.11 +/-  0.14 kohms
---------------------------------------
```

```
---------------------------------------
File FE556
relative RMS error = 0.030861
Ecorr = -228.06 millivolts
Icorr = 2.00665 +/- 0.05716 microamps/cm^2
ba    =  34.05 +/-   0.64 millivolts
bc    =  37.37 +/-   1.42 millivolts
Rp    =   3.86 +/-  0.11 kohms
---------------------------------------
```

```
---------------------------------------
File FE557
relative RMS error = 0.033995
Ecorr = -223.22 millivolts
Icorr = 2.31465 +/- 0.07973 microamps/cm^2
ba    =  36.55 +/-   1.11 millivolts
bc    =  39.10 +/-   1.79 millivolts
Rp    =   3.54 +/-  0.12 kohms
---------------------------------------
```

```
---------------------------------------
File FE558
relative RMS error = 0.032829
Ecorr = -219.41 millivolts
Icorr = 2.57045 +/- 0.08475 microamps/cm^2
ba    =  33.28 +/-   1.04 millivolts
bc    =  35.85 +/-   1.42 millivolts
Rp    =   2.92 +/-  0.09 kohms
---------------------------------------
```

```
----------------------------------------
File FE559
relative RMS error = 0.036933
Ecorr = -213.04 millivolts
Icorr = 3.80187 +/- 0.15673 microamps/cm^2
ba    =  36.98 +/-   1.95 millivolts
bc    =  42.86 +/-   2.09 millivolts
Rp    =   2.27 +/-  0.09 kohms
----------------------------------------
```

```
----------------------------------------
File FE5510
relative RMS error = 0.034996
Ecorr = -206.91 millivolts
Icorr = 3.48532 +/- 0.13455 microamps/cm^2
ba    =  35.82 +/-   1.59 millivolts
bc    =  35.87 +/-   1.66 millivolts
Rp    =   2.23 +/-  0.08 kohms
----------------------------------------
```

```
----------------------------------------
File FE5511
relative RMS error = 0.033866
Ecorr = -199.15 millivolts
Icorr = 2.66925 +/- 0.10185 microamps/cm^2
ba    =  30.41 +/-   0.98 millivolts
bc    =  31.05 +/-   1.43 millivolts
Rp    =   2.50 +/-  0.09 kohms
----------------------------------------
```

```
----------------------------------------
File FE5512
relative RMS error = 0.033131
Ecorr = -193.85 millivolts
Icorr = 2.68506 +/- 0.09301 microamps/cm^2
ba    =  33.46 +/-   0.76 millivolts
bc    =  32.35 +/-   1.49 millivolts
Rp    =   2.66 +/-  0.09 kohms
----------------------------------------
```

APPENDIX J

INSTALL.BAT: BATCH FILE THAT INSTALLS PCCORROS

```
@ECHO OFF
ECHO This batch file will install PCCORROS on drive C from
     any other drive.
ECHO The subdirectory PCCORROS will be created and all the
     files copied to it.
ECHO The batch file PCCORROS.BAT will be copied to the root
     directory.  To run
ECHO PCCORROS move to the root directory of drive C and type
     in "PCCORROS".
PAUSE
COPY \PCCORROS\PCCORROS.BAT  C:\
MD C:\PCCORROS
COPY \PCCORROS\*.*  C:\PCCORROS
C:
CD\
```

REFERENCES

1. Hanham, C.M., Gallagher, P.J., "CORROS: A Computer Program For Analyzing Polarization Resistance Data", DREA Technical Communication 86/303, April 1986.

2. National Instruments Corporation, MC-GPIB User Manual for the IBM Personal System/2 Computer Family with Micro Channel Architecture, Austin, Texas, March 1988.

3. International Business Machines Corporation, IBM Personal System/2 Model 50 Quick Reference, third ed., April 1988.

4. Princeton Applied Research Corp., Model 342 SOFTCORR Corrosion Measurement Software Operating Manual, Princeton, New Jersey, 1986.

5. International Business Machines Corporation, Disk Operating System Version 3.30: Reference, First ed., April 1987.

6. Princeton Applied Research Corp., HEAD START Creative Electrochemistry Software Preliminary Operating Manual, Princeton, New Jersey, 1986.

7. Forsythe, G.E., Malcolm, M.A. and Moler, C.B., "Computer Methods For Mathematical Computations", Prentice-Hall, Inc., Toronto, p. 76, 1977.

8. Borland International, TurboPASCAL Owner's Handbook Version 4.0, Scotts Valley, California, 1987.

9. American Society for Metals, EnPlot, Metals Park, Ohio, 1986.

10. Donahue, F.M., "Electrochemical Techniques in Corrosion Studies", Corrosion Chemistry, ACS Symposium Series 89, Brubaker, G.R. and Phipps, P.B.P., Eds., American Chemical Society, Washington, p. 52, 1979.

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence Research Establishment Atlantic	2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable) UNCLASSIFIED

3 TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title.)

IBM PC Analysis of Polarization Resistance Data

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Hardy, A.E. and Hanham, C.M.

5. DATE OF PUBLICATION (month and year of publication of document) July 1989	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.) 54	6b. NO. OF REFS (total cited in document) 10

6. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Communication

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Defence Research Establishment Atlantic

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant) Project No. 1AI	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)
10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.) DREA Technical Communication 89/307	10b. OTHER DOCUMENT NOS. (any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitations to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where futher distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

DCD03 2/06/87

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

13 ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both offical languages unless the text is bilingual).

An IBM PC has been set up to control an EG&G PARC M273 potentiostat to perform electrochemical corrosion experiments using PARC'S M342 corrosion measurement software. Modifications to the software permit the measurement of corrosion current over long periods of time using the Polarization Resistance measurement technique. A computer program, PCCORROS, is used to analyze the polarization resistance data to determine the polarization resistance, corrosion current, anodic and cathodic tafel constants, and the corrosion potential.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

polarization resistance

corrosion

electrochemical

computer program

IBM PC

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM