References

[1] -A. Sargent, R.P. Broadwater, J.C. Thompson, J. Nazarko, “Estimation of Diversity and KWhr-to-Peak-KW Factors from Load Research Data”, IEEE Transactions on Power Systems, Vol. 9, No. 3, August 1994.

[2] -A.K. Ghosh, D.L. Lubkeman, M..J. Downey, R.H. Jones, “Distribution Circuit State Estimation Using a Probabilistic Approach”, IEEE paper No. 96 WM 171-9 PWRS, IEEE Winter Power Meeting, 1996.

[3] -B. Borkowska, “Probabilistic Load Flow”, IEEE Transactions on PAS, Vol. PAS 93, No. 3, May 1974.

[4] -J.F. Dopazo, O.A. Klitin, A.M. Sasson, “Stochastic Load Flows”, IEEE Transactions on PAS, Vol. PAS 94, No. 2, March 1975.

[5] -R.D. Zimmerman, H-D. Chiang, “Fast Decoupled Power Flow for Unbalanced Radial Distribution Systems”, IEEE Transactions on Power Systems, Vol. 10, No. 4, November 1995.

[6] -B. Stott, O. Alsac, “Fast Decoupled Load Flow ”, IEEE Transactions on PAS, Vol. PAS 93, May 1974.

[7] -D. Rajicic, A. Bose, “A Modification to the Fast Decoupled Power Flow for Networks with High R/X Ratios”, IEEE Transactions on PAS, Vol. 3, No. 2, May 1988.

[8] -W.F. Tinney, C.E. Hart, “Power Flow Solution by Newton’s Method”, IEEE Transactions on PAS, Vol. PAS 86, No. 11, November 1967.

[9] -A.P. Sakis Meliopoulis, F. Zhang, “Multiphase Power Flow and State Estimation for Power Distribution Systems”, IEEE Transactions on Power Systems, Vol. 11, No. 2, May 1996.

[10] -A.G. Exposito, E.R. Ramos, “Radial Load Flow Technique for Radial Distribution Networks”, IEEE paper No. PE-344-PWRS-0-12-1996, IEEE Winter Power Meeting, 1997.

[11] -M.H. Haque, “Efficient Load Flow Method for Distribution Systems with Radial or Mesh Configuration”, IEE 1996.

[12] -M.E. Baran, A.W. Kelley, “Branch-Current Based State Estimation Method for Distribution Systems”, IEEE Summer Meeting 1994.

[13] -J.J. Burke, “Power Distribution Engineering: Fundamentals and Applications”, Marcel Dekker Inc., 1994.

[14] -A.M. Mood, F.A. Graybill, D.C. Boes, “Introduction to the Theory of Statistics”, Third Edition, McGraw-Hill, 1974.

[15] -B. Efron, R.J. Tibshirani, “An Introduction to the Bootstrap”, Chapman and Hall, 1993.

[16] -C.Z. Mooney, R.D. Duval, “Bootstrapping: a Nonparametric Approach to Statistical Inference”, Sage Publications, 1993.

[17] -J.S. Urban Hjorth, “Computer Intensive Statistical Methods”, Chapman and Hall, 1994.

[18] -A. Seppala, “The Estimation and Simulation of Electricity Customer Hourly Load Distribution”, 12th Power Systems Computation conference of Dresden, August 1996.

[19] -W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, “Numerical Recipes : The Art of Scientific Computing”, Cambridge University Press.

[20] -D.E. Knuth, “The Art of Computer Programming”, Seminumerical Algorithms, Vol. 2, 2nd ed., Addison-Wesley, 1981.

[21] -A.S.C. Ehrenberg, “Data Reduction-Analyzing and Interpreting Statistical Data”, J. Wiley and Sons, 1971.

[22] -W.N. Venables, B.D. Ripley, “Modern Applied Statistics with S-PLUS”, Springer-Verlag.

[23] -B.S. Everitt, “Statistical Analyses using S-PLUS”, Chapman and Hall, 1994.

Appendix A. Program *PROCESS 1*

==========

DATABASE MODIFICATION:

-<u>INITIAL BASE</u>: NUMERALS FOLLOW EACH OTHER WITHOUT BREAK. SEVERAL CUSTOMERS ONE AFTER ANOTHER.

- <u>FINAL BASE</u>: TABLES WHERE THE NUMBER OF COLUMNS LIES IN [1;10] AND WHERE EACH NUMBER CONSISTS OF 1 TO 10 NUMERALS. THERE WILL BE ONE TABLE PER FILE AND PER CUSTOMER.

==========

```
#include <string.h>
#include <stdio.h>
#include <process.h>
#include <stdlib.h>

const unsigned long M=52574;

void main(int argc, char * argv[])

        {

          char ch,fname[14],str[5];
          FILE *finput,*foutput;              //  2 pointers on the object FILE are defined.
                                              // (FILE is a structure model defined in stdio.h).
          int i, countlen=0,countcol=0;    // iterative variables.
          int lengroup=0;                  // "lentgroup" will be equal to the number of numerals
                                           // in each group.
          int column=0;                    // "column" will take the value of the number of columns.
          int temp=0;                     // buffer variable.
          unsigned long chcount=0L, fcount=0L;     // iterative variables.
```

```
if (argc < 3|| argc > 5)          // number of input variables is not right.
        {
         printf ("\nThe program should be used like this :");
         printf ("\n[Executive filename] [source filename]");
         printf (" [objective filename] [length of the number]");
         printf (" [number of columns]\n");
         exit (0);
        }


if ((finput=fopen(argv[1],"r"))==NULL)
        {
         printf ("Can't open file: %s\n",argv[1]);
         exit(0);
        }                                          // error on the input file.


if ((foutput=fopen(argv[2],"w"))==NULL)
        {
         printf ("Can't open file: %s\n",argv[2]);
         exit(0);
        }                                          // error on the output file.


if (strlen(argv[2])>5)
        {
         printf ("The length of the output file name should be less than 5");
         exit(0);
        }               //  the size of the name is reduced in order to add an expansion.


lengroup=6; column=10;            // values by lacking :
                                    // length of one number or group.
                                   // number of columns.


if (argc == 4 )                  // the first four input variables were entered
                                  // and the number of columns is 10 by lacking.
        {
         temp=atol (argv[3]);                  // chain [length of the number]
                                                 //  is converted in long integer.
```

```
        if (temp <= 0 || temp > 10)          // length of the number
                                             // is outside [1;10] interval.
                {
                 printf ("\nThe length of group is out of range and set to 6");
                }
        else
                {
                 lengroup= temp;         // length will be different from 6.
                }
       }


if (argc == 5)                          // all the input variables were entered.
       {
        temp=atol (argv[3]);         // for the chain [length of the number].

        if (temp <= 0 || temp > 10)
                {
                 printf ("\nThe length of group is out of range and set to 6");
                }
        else
                {
                 lengroup=temp;
                }


        temp=atol(argv[4]);     // for the chain [number of columns].

        if (temp <= 0 || temp > 10)
                {
                 printf ("\nThe number of columns is out of range and set to 10");
                }
        else
                {
                 column=temp;
                }
       }



fcount=1;
```

```
do
        {
         ultoa (fcount,str,10);
         strcpy (fname,argv[2]);    //   Define the name of the output file : it will consist
         strcat (fname,str);         // of the chain [str] + fcount (iterative number) +'.dat'.
         strcat (fname,".dat");      // for the first file : '[str]1.dat'.

         if ((foutput=fopen(fname,"w"))==NULL)
                 {
                  printf ("Can't open file: %s\n",fname);exit(0);
                 }

         chcount=0;

         for (i=0;i<12;i++)
                 {
                  fputc (fgetc(finput),foutput);       // read the 12 first numerals in the
                  chcount++;              // input file and write them in the output file.
                 }                        // 'chcount' is incremented.


         fprintf(foutput,"\n");            // the customer number is obtained.


         do
                 {
                  countcol = 0;

                  while (countcol < column)
                          {
                           countlen = 0;

                            while (countlen < lengroup)
                                   {
                                    ch = fgetc(finput);
                                    chcount++;

                                    if (ch =='\r'|| ch=='\t')
                                            continue;
                                    else if ((ch == EOF) || (chcount>= M))
                                            goto l1;
                                    else
                                            countlen++;
```

```
                    fputc(ch,foutput);
                }
```
// **for 'countcol' given**, **numerals are read in the input file . The value is copied in the output file in order to create a number of length 'lengroup'. If the end of the input file or the end of the customer data is reached, the process is stopped.**

```
            fprintf (foutput,"  ");
            countcol++;     // the number was billed and countcol was
                            // incremented in order to go to the next column.
            }

        l1: fprintf(foutput,"\n");
        }

    while ((ch !=EOF)&&(chcount<M));          // as long as the end of the input file
```
and the end of the customer data are not reached, go to the next line to repeat the process.

```
    fclose(fouput);           // the file '[str] fcount.dat' is closed. It represents
                              // the customer 'fcount'.

    fcount++;                 // 'fcount' is incremented for the next customer
                              // (customer 'fcount +1').
    }

  while (ch!=EOF);
  fclose(fintput);                // when the end of the input file
                                  // is reached, it is closed.
}
```

Appendix B. Extract from the '*H*eat *R*esidential' File

2000000000010055320057840062520069240037680018240050520053640033240050400030240122880083400091560122040024720040680118320058920063480041160017040017520055080077160060000047160067320060360036840074280009720028920027720027720027000028680043320040560032640027360027120050400028080027240044520026280028680044040027720037560045000029880024120042600026760024120023280025080025560025320025800356400254400116400110400158400104400734400398400262800126000112800112800216000291600092400134400108000144000145200106800100800098400094800154800322800462000652801076400549600310800660000325200108000124800093600306000420000524400331200450000380400684000286800271200326400376800237600252000608400300000477601290001048800542400736800606000654000625200369600549600619200571200416400328800452400454800292800450000276000444000304800260400558000338400297600573600450000762000772800627600294000110400102000466800450000637200092400327600358800661200450000362400318000412800414000246000501600738000840000769200319200244800141600129600103200316800307200328800379200294....

Doc 3.1 Data of the First Customer of the *H*eat *R*esidential Class
Before Processing by '*PROCESS 1*'

200000000001
005532 005784 006252 006924 003768 001824 005052 005364
003324 005040 003024 012288 008340 009156 012204 002472
004068 011832 005892 006348 004116 001704 001752 005508
007716 006000 004716 006732 006036 003684 007428 000972
002892 002772 002772 002700 002868 004332 004056 003264
002736 002712 005040 002808 002724 004452 002628 002868
004404 002772 003756 004500 002988 002412 004260 002676
002412 002328 002508 002556 002532 002580 003564 00 etc..

Doc 3.2 Data of the First Customer of the *H*eat *R*esidential Class
After Processing by '*PROCESS 1*'

Appendix C. Program *PROCESS 2*

```
======================================================================

CREATE THE FILE HCIin.DAT. THIS FILE WILL BE USED TO CALCULATE
      CONFIDENCE INTERVALS FOR NODAL HOURLY POWER CONSUMPTION,
      FOR A NODE (N1,N2,N3 ARE KNOWN) AND A PARTICULAR HOUR OF THE
      YEAR GIVEN.

======================================================================


=========Variables declaration

      INTEGER*4 I, J, K, L, N1, N2, N3, d, h
      DOUBLE PRECISION VAL
      REAL*4 CONS1, CONS2, CONS3, CONStot

=========Arrays declaration

      REAL*4 TRASH, Mbuf (1095,8), BUF1(139), BUF2(158), BUF3(180)

=========Characters declaration

      CHARACTER PREF1*2 /'ht'/, PREF2*2 /'nh'/, PREF3*2 /'cl'/, Nb1(9)*1
      CHARACTER Nb2(90)*2, Nb3(101)*3, FIN*4 /'.DAT'/, NAME1*9
      CHARACTER NAME2*9, NAME3*9

      DATA (Nb1(I), I=1, 9) /       '1','2','3','4','5','6','7','8','9' /

      DATA (Nb2(I), I=1,90) /       '10','11','12','13','14',
     &                        '15','16','17','18','19','20','21','22','23',
     &                        '24','25','26','27','28','29','30','31','32',
     &                        '33','34','35','36','37','38','39','40','41',
     &                        '42','43','44','45','46','47','48','49','50',
     &                        '51','52','53','54','55','56','57','58','59',
     &                        '60','61','62','63','64','65','66','67','68',
     &                        '69','70','71','72','73','74','75','76','77',
     &                        '78','79','80','81','82','83','84','85','86',
```

```
     &                    '87','88','89','90','91','92','93','94','95',
     &                    '96','97','98','99' /

          DATA (Nb3(I), I=1,101)/      '100','101','102','103','104',
     &                    '105','106','107','108','109','110','111','112','113',
     &                    '114','115','116','117','118','119','120','121','122',
     &                    '123','124','125','126','127','128','129','130','131',
     &                    '132','133','134','135','136','137','138','139','140',
     &                    '141','142','143','144','145','146','147','148','149',
     &                    '150','151','152','153','154','155','156','157','158',
     &                    '159','160','161','162','163','164','165','166','167',
     &                    '168','169','170','171','172','173','174','175','176',
     &                    '177','178','179','180','181','182','183','184','185',
     &                    '186','187','188','189','190','191','192','193','194',
     &                    '195','196','197','198','199','200' /
```

=========**Read the input variables**

```
          WRITE (*,1100)
1100      FORMAT ('Enter the number of measurements for Heat Residential',
     &           'Customers :',\)
          READ (*,*) N1

          WRITE (*,1200)
1200      FORMAT ('Enter the number of measurements for Non-Heat',
     &          'Residential Customers :',\)
          READ (*,*) N2

          WRITE (*,1300)
1300      FORMAT ('Enter the number of measurements for Commercial',
     &           ' Load :',\)
          READ (*,*) N3

          WRITE (*,1400)
1400      FORMAT ('Enter an integer.',
     &            'It will be used to generate random samples :',\)
          READ (*,*) VAL

          WRITE (6,*)'Enter the day of the year (1 to 365) ?'
          READ (6,*) d

          WRITE (6,*)'Enter the hour in this day ?'
```

```
      READ (6,*)  h
```

=========**Main Program**==========

```
      OPEN (40, FILE='HCIin.DAT')

      DO L=1,100
```

=========**Select randomly and without replacement N1,N2,N3 customers**

```
      CALL RANDONE (139,VAL,N1,BUF1)
      CALL RANDONE (158,VAL,N2,BUF2)
      CALL RANDONE (180,VAL,N3,BUF3)
```

=========**Pick up the hourly consumption for each customer**

```
      CONS1=0.

      DO K=1, N1

            IF (BUF1(K).LT.10) THEN
                  NAME1=PREF1//Nb1(BUF1(K))//FIN
            ELSE IF (BUF1(K).GE.10.AND.BUF1(K).LT.100) THEN
                  NAME1=PREF1//Nb2(BUF1(K)-9)//FIN
            ELSE
                  NAME1=PREF1//Nb3(BUF1(K)-99)//FIN
            ENDIF

            OPEN (10, FILE=NAME1, status='old')
            READ (10,*) TRASH

            DO I=1,1095
                  READ(10,*) (Mbuf(I,J),J=1,8)
            ENDDO

            CLOSE (10)

            IF (h.LE.8) THEN
                  CONS1=CONS1+Mbuf((3*d-2),h)
```

```
        ELSE IF (h.GT.8.AND.h.LE.16) THEN
                CONS1=CONS1+Mbuf((3*d-1),h)
        ELSE IF (h.GT.16) THEN
                CONS1=CONS1+Mbuf(3*d,h)
        ENDIF

ENDDO

CONS2=0.

DO K=1, N2

        IF (BUF2(K).LT.10) THEN
                NAME2=PREF2//Nb1(BUF2(K))//FIN
        ELSE IF (BUF2(K).GE.10.AND.BUF2(K).LT.100) THEN
                NAME2=PREF2//Nb2(BUF2(K)-9)//FIN
        ELSE
                NAME2=PREF2//Nb3(BUF2(K)-99)//FIN
        ENDIF

        OPEN (20, FILE=NAME2, status='old')
        READ(20,*)TRASH

        DO I=1,1095
                READ(20,*) (Mbuf(I,J),J=1,8)
        ENDDO

        CLOSE (20)

        IF (h.LE.8) THEN
                CONS2=CONS2+Mbuf((3*d-2),h)
        ELSE IF (h.GT.8.AND.h.LE.16) THEN
                CONS2=CONS2+Mbuf((3*d-1),h)
        ELSE IF (h.GT.16) THEN
                CONS2=CONS2+Mbuf(3*d,h)
        ENDIF

ENDDO

CONS3=0.

DO K=1, N3
```

```
            IF (BUF3(K).LT.10) THEN
                  NAME3=PREF3//Nb1(BUF3(K))//FIN
            ELSE IF (BUF3(K).GE.10.AND.BUF3(K).LT.100) THEN
                  NAME3=PREF3//Nb2(BUF3(K)-9)//FIN
            ELSE
                  NAME3=PREF3//Nb3(BUF3(K)-99)//FIN
            ENDIF

            OPEN (30, FILE=NAME3, status='old')
            READ(30,*) TRASH

            DO I=1,1095

                  READ(30,*) (Mbuf(I,J),J=1,8)
            ENDDO

            CLOSE (30)

            IF (h.LE.8) THEN
                  CONS3=CONS3+Mbuf((3*d-2),h)
            ELSE IF (h.GT.8.AND.h.LE.16) THEN
                  CONS3=CONS3+Mbuf((3*d-1),h)
            ELSE IF (h.GT.16) THEN
                  CONS3=CONS3+Mbuf(3*d,h)
            ENDIF

      ENDDO
```

=========**Calculate the nodal hourly power consumption**

```
      CONStot=CONS1+CONS2+CONS3
      WRITE (40,*) CONStot


      ENDDO

      CLOSE (40)

      STOP
      END
```

```
======================================================================

      SUBROUTINE RANDONE (MAXI,VAL,N,BUF)

======================================================================
```

Create a sample of N values, randomly without replacement

```
      IMPLICIT LOGICAL (A-Z)
      INTEGER I,N,CHECK(200), BUFFER, MAXI
      REAL*4 BUF (200)
      DOUBLE PRECISION A, C, M, VAL, X
      M = 2.D0**32
      A = 3141592621.D0
      C = 1.D0
      X=DMOD(VAL+C,M)

      DO I=1, 200
            CHECK(I)=0
       ENDDO

      DO I=1,N

10          X=DMOD( A*X+C,M)
            BUFFER=1+DABS(X/M)*MAXI
```

***A random number was generated. It lies in [1;MAXI].**
***If a value is repeated, the loop below will permit to delete it and choose another one.**

```
            IF (CHECK(BUFFER).EQ.0) THEN
                  CHECK (BUFFER)=1
            ELSE
                  GOTO 10
            ENDIF

            BUF(I)=BUFFER

      ENDDO

      VAL=X
      RETURN
      END
```

Appendix D. Program *PROCESS 3*

```
=================================================================
```

GATHER MAXIMA VALUES OF POWER CONSUMPTION PER CUSTOMER INSIDE THE SAME FILE, FOR A MONTH AND A CLASS GIVEN.

```
=================================================================
```

=========**Variables declaration**

```
      INTEGER*4 I, J, N, K
      REAL*4 TRASH, Mbuf (1095,8), MAXIMUM
      CHARACTER PREF1*2, Nb1(9)*1, Nb2(90)*2, Nb3(101)*3,
     &          FIN*4 /'.DAT'/,NAME1*9


      DATA (Nb1(I), I=1,9)/     '1','2','3','4','5','6','7','8','9' /

      DATA (Nb2(I), I=1,90)/    '10','11','12','13','14',
     &                          '15','16','17','18','19','20','21','22','23',
     &                          '24','25','26','27','28','29','30','31','32',
     &                          '33','34','35','36','37','38','39','40','41',
     &                          '42','43','44','45','46','47','48','49','50',
     &                          '51','52','53','54','55','56','57','58','59',
     &                          '60','61','62','63','64','65','66','67','68',
     &                          '69','70','71','72','73','74','75','76','77',
     &                          '78','79','80','81','82','83','84','85','86',
     &                          '87','88','89','90','91','92','93','94','95',
     &                          '96','97','98','99'/

      DATA (Nb3(I), I=1,101)/   '100','101','102','103','104',
     &                          '105','106','107','108','109','110','111','112','113',
     &                          '114','115','116','117','118','119','120','121','122',
     &                          '123','124','125','126','127','128','129','130','131',
     &                          '132','133','134','135','136','137','138','139','140',
     &                          '141','142','143','144','145','146','147','148','149',
     &                          '150','151','152','153','154','155','156','157','158',
```

```
     &                          '159','160','161','162','163','164','165','166','167',
     &                          '168','169','170','171','172','173','174','175','176',
     &                          '177','178','179','180','181','182','183','184','185',
     &                          '186','187','188','189','190','191','192','193','194',
     &                          '195','196','197','198','199','200' /
```

========**Main Program**==========

```
      WRITE (*,10)
10    FORMAT ('Enter the file pref ?'\)
      READ (*,'(A)') PREF1
      WRITE (6,*) 'Enter the number of file ?'
      READ (6,*) N

      OPEN (20,FILE=PREF1//'jan.dat')
      OPEN (30,FILE=PREF1//'feb.dat')
      OPEN (40,FILE=PREF1//'mar.dat')
      OPEN (50,FILE=PREF1//'apr.dat')
      OPEN (60,FILE=PREF1//'may.dat')
      OPEN (70,FILE=PREF1//'jun.dat')
      OPEN (80,FILE=PREF1//'jul.dat')
      OPEN (90,FILE=PREF1//'aug.dat')
      OPEN (100,FILE=PREF1//'sep.dat')
      OPEN (110,FILE=PREF1//'oct.dat')
      OPEN (120,FILE=PREF1//'nov.dat')
      OPEN (130,FILE=PREF1//'dec.dat')

      DO K=1, N

            IF (K.LT.10) THEN
                  NAME1=PREF1//Nb1(K)//FIN
            ELSE IF (K.GE.10.AND.K.LT.100) THEN
                  NAME1=PREF1//Nb2(K-9)//FIN
            ELSE
                  NAME1=PREF1//Nb3(K-99)//FIN
            ENDIF

            OPEN (140, FILE=NAME1, status='old')
            READ(140,*) TRASH
```

```
DO I=1,1095
        READ(140,*) (Mbuf(I,J),J=1,8)
ENDDO

CLOSE (140)
 MAXIMUM=0.

DO I=1,93
         DO J=1,8
                 MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (20,*) MAXIMUM
MAXIMUM=0.

DO I=94,177
        DO J=1,8
                MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (30,*) MAXIMUM
MAXIMUM=0.

DO I=178,270
        DO J=1,8
                MAXIMUM=AMAX1 (MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (40,*) MAXIMUM
MAXIMUM=0.

DO I=271,360
        DO J=1,8
                MAXIMUM=AMAX1 (MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE(50,*) MAXIMUM
MAXIMUM=0.
```

```
DO I=361,453
        DO J=1,8
                MAXIMUM=AMAX1 (MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (60,*) MAXIMUM
MAXIMUM=0.

DO I=454,543
        DO J=1,8
                MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (70,*) MAXIMUM
MAXIMUM=0.
DO I=544,636
        DO J=1,8
                MAXIMUM=AMAX1 (MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (80,*) MAXIMUM
MAXIMUM=0.

DO I=637,729
        DO J=1,8
                MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (90,*) MAXIMUM
MAXIMUM=0.

DO I=730,819
        DO J=1,8
                MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
        ENDDO
ENDDO

WRITE (100,*) MAXIMUM
MAXIMUM=0.
```

```
        DO I=820,912
                DO J=1,8
                        MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
                ENDDO
        ENDDO

        WRITE (110,*) MAXIMUM
        MAXIMUM=0.

        DO I=913,1002
                DO J=1,8
                        MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
                ENDDO
        ENDDO

        WRITE (120,*) MAXIMUM
        MAXIMUM=0.

        DO I=1003,1095
                DO J=1,8
                        MAXIMUM=AMAX1(MAXIMUM,Mbuf(I,J))
                ENDDO
        ENDDO

        WRITE (130,*) MAXIMUM

ENDDO

CLOSE(20)
CLOSE(30)
CLOSE(40)
CLOSE(50)
CLOSE(60)
CLOSE(70)
CLOSE(80)
CLOSE(90)
CLOSE(100)
CLOSE(110)
CLOSE(120)
CLOSE(130)

STOP
END
```

Appendix E. Random Generator

==========

This basic program generates numbers randomly with replacement. By adding a simple loop from this first program, it is then possible to generate numbers randomly without replacement.

==========

=========**Variables declaration**

```
      IMPLICIT LOGICAL (A-Z)
      INTEGER I, N, CHECK(200), BUFFER, MAXI
      REAL*4 BUF(200)
      DOUBLE PRECISION A, C,M , VAL, X

      M = 2.D0**32
      A = 3141592621.D0
      C = 1.D0

      X=DMOD(VAL+C,M)
```

=========**Main Program**==========

```
      DO I=1,200
            CHECK(I)=0
      ENDDO


      DO I=1,N
10          X=DMOD( A*X+C,M)
            BUFFER=1+DABS(X/M)*MAXI
```

***A random number was generated. It lies in [1;MAXI]. This program generates numbers randomly with replacement.**

***If a value is repeated, the loop below will permit to delete it and to choose another one. If this part is added to the previous program, random numbers are generated without replacement.**

```
            IF (CHECK(BUFFER).EQ.0) THEN
                  CHECK (BUFFER)=1
            ELSE
                  GOTO 10
            ENDIF

      ENDDO

      VAL=X

      RETURN
      END
```

Appendix F. Confidence Interval Estimation for Nodal Hourly Power Consumption by using the Nonparametric Bootstrap Method

==========

-Let us consider hourly data recorded during several years for a node and an hour of the year given (file HCIin).

-A 95% confidence interval for nodal hourly power consumption, for this hour & this node, will be inferred by using the nonparametric bootstrap method.

==========

=========**Variables declaration**

```
      INTEGER*4 N, I, J
      REAL*4 MAUDE, small5, BUF5
      DOUBLE PRECISION VAL
```

=========**Arrays declaration**

```
      REAL*4 SAMPLE(100),RESAMPLE(100),RES(2000)
```

=========**Read the input variables**

```
      WRITE (*,1100)
1100  FORMAT ('Enter an integer.',
     &        'It will be used to generate random samples :',\)
      READ (*,*) VAL

      WRITE (*,1200)
1200  FORMAT (‘Enter the number of customers for the node:’,\)
      READ (*,*) N
```

=========**Main Program**==========

```
OPEN (10, FILE='HCIin.DAT')
READ (10,*) (SAMPLE(I),I=1,100)
CLOSE (10)

OPEN (20, FILE='HCIout1.DAT')
OPEN (30, FILE='HCIout2.DAT')
```

=========**Resamples**

Create 2000 resamples of N measurements. Infer for each resample the value of the mode :

```
DO I=1,2000
        CALL RANDTWO (VAL,N,SAMPLE,RESAMPLE)
        CALL MODE (N,RESAMPLE,MAUDE)
        RES(I)=MAUDE
        WRITE(30,*) RES(I)
ENDDO
```

=========**Find a 95% confidence interval for the mode**

```
DO I=1,1999

        small5=RES(I)

        DO J=I+1,2000
                small5=MIN (small5,RES(J))
        ENDDO

        DO J=I+1,2000
                IF (small5.EQ.RES(J)) THEN
                BUF5=RES(I)
                RES(I)=RES(J)
                RES(J)=BUF5
                ENDIF
        ENDDO

ENDDO
WRITE (20,*)' MODE ESTIMATOR'
```

```
      WRITE (20,*) '95%-CONFIDENCE INTERVAL: [',RES(50),';',RES(1950),']'

      CLOSE(20)
      CLOSE(30)

      STOP
      END
```

==========

SUBROUTINES

==========

SUBROUTINE RANDTWO (VAL,N,SAMPLE,RESAMPLE)

Select N values among N values, with replacement

```
      IMPLICIT LOGICAL (A-Z)
      INTEGER I,N,RandNb
      REAL*4 SAMPLE(100),RESAMPLE(100)
      DOUBLE PRECISION A,C,M,VAL,X

      M = 2.D0**32
      A = 3141592621.D0
      C = 1.D0
      X=DMOD(VAL+C,M)

      DO I=1,N
            X=DMOD(A*X+C,M)
            RandNb=1+DABS(X/M)*N
```

***A random number was generated. It lies in [1;N].**

```
            RESAMPLE(I)=SAMPLE(RandNb)
      ENDDO

      VAL=X
```

```
      RETURN
      END
```

==========

```
SUBROUTINE MODE (N,RESAMPLE,MAUDE)
```

Calculate the mode for a resample given

```
      REAL*4 RESAMPLE(100),small2,BUF2,CLASS,BUF3(100)
      REAL*4 BUF4(100),MAUDE
      INTEGER I,J,K1,K2,N,Nclass
```

***Arrange data in increasing order inside RESAMPLE**

```
      DO I=1,N-1

            small2=RESAMPLE(I)

            DO J=I+1,N
                  small2=MIN (small2,RESAMPLE(J))
            ENDDO

            DO J=I+1,N
                  IF(small2.EQ.RESAMPLE(J)) THEN
                        BUF2=RESAMPLE(I)
                        RESAMPLE(I)=RESAMPLE(J)
                        RESAMPLE(J)=BUF2
                  ENDIF
            ENDDO

      ENDDO
```

***Calculate the maximum frequency class**

```
      Nclass=1100
      CLASS=(RESAMPLE(N)-RESAMPLE(1))/Nclass
      K2=0

      DO I=1,Nclass

            K1=0
            DO J=1,N
                  IF(RESAMPLE(J).GE.(RESAMPLE(1)+(I-1)*CLASS).AND.
     &                        RESAMPLE(J).LE.(RESAMPLE(1)+I*CLASS)) THEN
                        K1=K1+1
                        BUF3(K1)=RESAMPLE(J)
                  ENDIF
            ENDDO
            IF(K1.GT.K2) THEN
                  DO J=1,K1
                        BUF4(J)=BUF3(J)
                  ENDDO
                  K2=K1
            ENDIF

      ENDDO
```

***Calculation of the MODE**

```
      IF(MOD(K2,2).EQ.0) THEN
            MAUDE=(BUF4(K2/2)+BUF4((K2/2)+1))/2
      ELSE
            MAUDE=BUF4((K2+1)/2)
      ENDIF

      RETURN
      END
```

Appendix G. Confidence Interval Estimation for Maximum Power Consumption per Customer by using the Parametric Bootstrap Method

Part I

==========

-N1 customers from the class1, N2 from the class2, N3 from the class3 make up the node.

-For a month given, a sample is created by selecting N1 customers, randomly without replacement, from measured customers group 1, N2...

-For each customer of this sample, his maximum power consumption for the period is isolated. These N1+N2+N3 measurements form the basic sample for the parametric bootstrap study.

-Parameters for the beta, the lognormal and the chi-square distributions are calculated.

==========

=========**Variables declaration**

```
INTEGER*4 I, N1, N2, N3, N
REAL*4 SAMPLmin, SAMPLmax, MEAN,VAR, PARa, PARb
REAL*4 PARmu, PARsigma, small, BUF
DOUBLE PRECISION VAL
CHARACTER*3 PREF
```

=========**Arrays declaration**

```
REAL*4 SAMPLE1(139), SAMPLE2(158), SAMPLE3(180), SAMPLE(500)
REAL*4 BUF1(139), BUF2(158), BUF3(180), DISTRIB(500)
```

=========**Read the input variables**

```
      WRITE (*,1001)
1001  FORMAT ('Enter the month by its 3 first letters :',\)
      READ (*,'(A)') PREF
      WRITE (*,1100)
1100  FORMAT ('Enter the number of measurements for Heat Residential',
     &        'Customers :',\)
      READ (*,*) N1

      WRITE (*,1200)
1200  FORMAT ('Enter the number of measurements for Non-Heat',
     &        'Residential Customers :',\)
      READ (*,*) N2

      WRITE (*,1300)
1300  FORMAT ('Enter the number of measurements for Commercial',
     &        ' Load :',\)
      READ (*,*) N3

      WRITE (*,1400)
1400  FORMAT ('Enter an integer.',
     &    'It will be used to generate random samples :',\)
      READ (*,*) VAL
```

=========**Main Program**==========

```
      OPEN (10, FILE='ht'//PREF//'.DAT')
      OPEN (20, FILE='nh'//PREF//'.DAT')
      OPEN (30, FILE='cl'//PREF//'.DAT')
      READ (10,*) (SAMPLE1(I),I=1,139)
      READ (20,*) (SAMPLE2(I),I=1,158)
      READ (30,*) (SAMPLE3(I),I=1,180)

      CLOSE (10)
      CLOSE (20)
      CLOSE (30)

      OPEN (40, FILE='MCI3PARA.DAT')
```

```
      OPEN (50, FILE='MCI3DIST.DAT')
```

=========**Create the basic sample**

```
      CALL RANDONE (139,VAL,N1,SAMPLE1,BUF1)
      CALL RANDONE (158,VAL,N2,SAMPLE2,BUF2)
      CALL RANDONE (180,VAL,N3,SAMPLE3,BUF3)

      N=N1+N2+N3

      WRITE(40,*)' '
      WRITE(40,*)' '
      WRITE(40,*)'MONTH OF:   ',PREF,';'
      WRITE(40,*)' '
      WRITE(40,*)' '
      WRITE(40,*)'HR=',N1,';'
      WRITE(40,*)'NHR=',N2,';'
      WRITE(40,*)'CL=',N3,';'
      WRITE(40,*)' '
      WRITE(40,*)' '
      WRITE(40,*)' '

      DO I=1,N1
            SAMPLE(I)=BUF1(I)
      ENDDO

      DO I=1,N2
            SAMPLE(I+N1)=BUF2(I)
      ENDDO

      DO I=1,N3
            SAMPLE(I+N1+N2)=BUF3(I)
      ENDDO
```

=========**Infer the distributions' parameters**

```
      SAMPLmin=SAMPLE(1)
      SAMPLmax=SAMPLE(1)

      DO I=1,N
            SAMPLmin=AMIN1(SAMPLmin,SAMPLE(I))
            SAMPLmax=AMAX1(SAMPLmax,SAMPLE(I))
      ENDDO
```

```
MEAN=0.
VAR=0.

DO I=1,N
        DISTRIB(I)=(SAMPLE(I)-SAMPLmin)/(SAMPLmax-SAMPLmin)
        MEAN=MEAN+DISTRIB(I)
ENDDO

MEAN=MEAN/FLOAT(N)

DO I=1,N
        VAR=VAR+(DISTRIB(I)-MEAN)*(DISTRIB(I)-MEAN)
ENDDO

VAR=VAR/FLOAT(N-1)

WRITE(40,*)'NORMALIZED SCALE'
WRITE(40,*)' '
WRITE(40,*)'SAMPLmin=',SAMPLmin,';'
WRITE(40,*)'SAMPLmax=',SAMPLmax,';'
WRITE(40,*)' '
WRITE(40,*)'MEAN=',MEAN,';'
WRITE(40,*)'VAR=',VAR,';'
WRITE(40,*)' '
WRITE(40,*)'VAL=',VAL,';'
WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)'PARAMETERS FOR EACH DISTRIBUTION'
WRITE(40,*)' '
WRITE(40,*)' '
```

***Beta distribution**

```
PARa=(MEAN/VAR)*(MEAN*(1.-MEAN)-VAR)
PARb=((1.-MEAN)/VAR)*(MEAN*(1.-MEAN)-VAR)

WRITE(40,*)'BETA DISTRIBUTION'
WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)'PARa=',PARa,';'
WRITE(40,*)'PARb=',PARb,';'
```

*__Lognormal distribution__

```
PARmu=(ALOG((MEAN**4)/(VAR+MEAN*MEAN)))/2.
PARsigma=SQRT(ALOG((VAR+MEAN*MEAN)/(MEAN*MEAN)))

WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)'LOGNORMAL DISTRIBUTION'
WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)'PARmu=',PARmu,';'
WRITE(40,*)'PARsigma=',PARsigma,';'
```

*__Chi-square distribution__

```
degfree=MEAN

WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)'CHI-SQUARE DISTRIBUTION'
WRITE(40,*)' '
WRITE(40,*)' '
WRITE(40,*)'DEGREE of FREEDOM=',degfree,';'
```

*__Arrange data in increasing order inside the distribution__

```
DO I=1,N-1
        small=DISTRIB(I)

        DO J=I+1,N
                small=MIN (small,DISTRIB(J))
        ENDDO

        DO J=I+1,N
                IF(small.EQ.DISTRIB(J)) THEN
                        BUF=DISTRIB(I)
                        DISTRIB(I)=DISTRIB(J)
                        DISTRIB(J)=BUF
                ENDIF
        ENDDO
ENDDO
```

```
        DO I=1,N
              WRITE(50,*)DISTRIB(I)
        ENDDO

        CLOSE(40)
        CLOSE(50)

        STOP
        END
```

==========

SUBROUTINE

==========

SUBROUTINE RANDONE (MAXI,VAL,N,SAMP,BUF)

Create a sample of N values randomly, without replacement

```
        IMPLICIT LOGICAL (A-Z)
        INTEGER I,N,CHECK(200),BUFFER,MAXI
        REAL*4 SAMP(200),BUF(200)
        DOUBLE PRECISION A,C,M,VAL,X

        M = 2.D0**32
        A = 3141592621.D0
        C = 1.D0
        X=DMOD(VAL+C,M)

        DO I=1,200
              CHECK(I)=0
        ENDDO

        DO I=1,N
10            X=DMOD( A*X+C,M)
              BUFFER=1+DABS(X/M)*MAXI

              IF (SAMP(BUFFER).EQ.0.) THEN
                    GOTO 10
```

```
        ENDIF

        IF(CHECK(BUFFER).EQ.0) THEN
            CHECK(BUFFER)=1
        ELSE
            GOTO 10
        ENDIF

        BUF(I)=SAMP(BUFFER)

ENDDO

VAL=X

RETURN
END
```

Appendix H. Confidence Interval Estimation for Maximum Power Consumption per Customer by using the Parametric Bootstrap Method

Part II

==========

1.KULLBACK-LEIBLER'S MEASURE

=========**Density distribution for the Beta, Lognormal and Chi-square laws**

```
spbeta <-   dbeta (madist, a, b)
splogn <-  dlnorm (madist, mu, sigma)
spchisq <- dchisq (madist, degfree)
```

where 'madist' is the vector made up of the sample's values.
(a,b), (mu, sigma), (degfree) are the parameters associated with the 3 distributions.

=========**K-L's measure**

```
KULLBACK <- function(x)
{
        KL <- 0
        for(i in 1:length(x))
        {
                KL <- KL + log(x[i])
        }
        KL <- KL/length(x)
        KL
}
```

Kullback-Leibler's measure is obtained for a particular density. This density is the input parameter 'x'.

2.PARAMETRIC BOOTSTRAP

=========**Relative frequency distribution of the maximum likelihood**

```
maxboot <- numeric (2000)

for ( i in 1:2000) maxboot [i] <- max ( rlnorm (N1+N2+N3, mu, sigma))

for ( i in 1:2000) maxboot [i] <- maxboot[i] * ( SAMPLmax-SAMPLmin) + SAMPLmin
```

2000 values of maximum likelihood were calculated for 2000 samples drawn randomly from the lognormal distribution. The distribution is the closest to the empirical distribution and it is chosen by applying the K-L's measure.

(N1+N2+N3) are the number of customers for the node considered.

=========**95% Confidence interval**

```
quantile (maxboot, c (0.025, 0.975))
```

Pierre Cugnet

The author was born on November 23, 1972 and grew up near Marseille, France. He attended the Ecole Nationale Superieure d'Ingenieurs Electriciens de Grenoble where he received a diploma in electrical engineering in 1996. In the spring of 1996, the author enrolled in the graduate school at Virginia Polytechnic Institute and State University as an M.S. student in electrical engineering.