```
@
  PROGRAM TO MAXIMIZE THE SMSE OBJECTIVE FUNCTION BY SIMULATED ANNEALING

  REFERENCE:  H. SZU AND R. HARTLEY (1987).  FAST SIMULATED ANNEALING,
           PHYSICS LETTERS A, 12, 157-162.

  THREE GAUSS PROCEDURES NAMED KFCN, DKFCN AND DDKFCN FOLLOW THIS MAIN
  PROGRAM.  THEY MUST BE STORED IN SEPARATE FILES BEFORE ATTEMPTING TO
  RUN THE PROGRAM.
@
  NEW;
  TSTRT = DATE;
  OUTPUT FILE = CSMSF.OUT ON; @ **** CREATE A FILE FOR OUTPUT **** @
  OUTPUT OFF;

  DATASET = "C:\\GAUSS\\SUBDIR\\FILE"; @ **** IDENTIFY INPUT DATA **** @

@
  READ THE DATA AND SET UP ARRAYS
@
  OPEN FY=^DATASET;
  N = ROWSF(FY);
  YZ=READR(FY,N);
  CLOSE(FY);

@ ***** IDENTIFY AND STANDARDIZE THE CONTINUOUS INDEPENDENT VARIABLES ***** @

   QNTL = 0.25;
   XVAR = YZ[.,1:3];
   temp = stdc(xvar);
   XVAR = (XVAR - MEANC(XVAR)')./STDC(XVAR)';
   XVAR = ONES(N,1)~YZ[.,5]~XVAR;

@ ***** IDENTIFY THE DEPENDENT VARIABLE ***** @

  YVAR = YZ[.,4];
  SY = 2*(YVAR .== 1) - 2*(1 - QNTL);
  CLEAR YZ;

  n = rows(xvar);   @ number of observations @
  dx = cols(xvar);      @ dimension of x @

@
  SPECIFY STARTING VALUES OF PARAMETERS OF THE MODEL
@
  LET X0[4,1]  =  0  1.732  0.032  0.346;
  LET YSCALE[4,1] = 1 1 0.25 0.25; @ *** SCALE PARAM. FOR CAUCHY DIST. *** @
  LET PARNAME[4,1] = INTRCPT  DEM  EMPLU  CAP;
@
  SET PARAMETERS OF THE METHOD
@
  QNTL = 0.5;
  K = 4;  @ ******* NO. OF VARIABLES IN THE OPTIMIZATION ******* @
```

```
  KPL = K + 1;
  MAXI = 200;  @  ******  MAXIMUM NUMBER OF ANNEALING ITERATIONS *****  @
  MAXINR = 50; @ ***** MAXIMUM NUMBER OF NEWTON-RAPHSON INTERATIONS ****** @
  M = 1;  @ ******* NO. OF ITERATIONS AT EACH TEMPERATURE ******* @
  T0 = .02;  @ ******* INITIAL TEMPERATURE ******* @
  SIG = 0.33;  @ ****** SET BANDWIDTH ****** @
  H = 2;       @ ****** SET ORDER OF KERNEL ******* @
  DEL = 0.1;   @ ****** DELTA VALUE FOR BIAS COMPUTATION ******* @
  PSTOP = 0.0000001;  @ ******* STOPPING CRITERION ******** @
  RPT = 1;
  DO UNTIL RPT > 2;
  OUTPUT ON;
  "     SMOOTHED MAXIMUM SCORE ESTIMATION BY SIMULATED ANNEALING";
  "               METHOD OF SZU AND HARTLEY     ";
  "";
  "";
  "DATASET = ";; DATASET;
  "NUMBER OF OBSERVATIONS: ";; N;
  "QUANTILE:  ";; QNTL;
  "ORDER OF KERNEL: ";; H;
  "BANDWIDTH :";; SIG;
  "DELTA: ";; DEL;
  "OUTER ITERATIONS = ";; MAXI;
  "NEWTON-RAPHSON ITERATIONS = ";; MAXINR;
  "STOPPING CRITERION = ";; PSTOP;
@
  SELECT THE INITIAL TEMPERATURE BY SAMPLING
@

/*
  SMPL = 25;
  SCUM = ZEROS(SMPL,1);
  ISAM = 1;
  DO UNTIL ISAM > SMPL;
     X = X0 + 30*(RNDU(K,1) - 0.5);
     ARG = XVAR[.,1:K]*X + XVAR[.,KPL];
     PHI0 = SY.*KFCN(ARG/SIG,N,H);
     PHI0 = -MEANC(PHI0);
     SCUM[ISAM,.] = PHI0;
     ISAM = ISAM + 1;
     ENDO;
  T0 = (STDC(SCUM))^2;
*/

  "INITIAL TEMPERATURE = ";; T0;
  "";
  OUTPUT OFF;
  SCUM = ZEROS(MAXI,1);
@
  COMPILE PROCEDURES
@
  #INCLUDE KFCN.PRC;
  #INCLUDE DKFCN.PRC;
  #INCLUDE DDKFCN.PRC;
```

```
@
 START A LOOP TO IMPLEMENT THE ALGORITHM
@
 BX = X0;
 IRET = 0;
 ANNL:
 ITER = 1;
 DO UNTIL ITER > MAXI;
@
 RESET TEMPERATURE
@
   T = T0/(100*IRET + ITER);
    ARG = XVAR[.,1:K]*BX + XVAR[.,KPL];
    PHI0 = SY.*KFCN(ARG/SIG,N,H);
    PHI0 = -MEANC(PHI0);
    IF ITER == 1;
     PHIOPT = PHI0;
     XOPT = BX;
     ENDIF;
@
 GENERATE CAUCHY-DISTRIBUTED Y'S;
@
 Y = RNDU(K,1);
 Y = T*TAN(PI*(Y - 0.5));
 Y = Y.*YSCALE;
@
 GENERATE STEP
@
 XS = BX + Y;
 ARG = XVAR[.,1:K]*XS + XVAR[.,KPL];
 PHI1 = SY.*KFCN(ARG/SIG,N,H);
 PHI1 = -MEANC(PHI1);
 DPHI = PHI1 - PHI0;
 IF DPHI <  0;
   BX = XS;
   PHI0 = PHI1;
   IF PHIOPT > PHI1;
    PHIOPT = PHI1;
    XOPT = XS;
    ENDIF;
    IREJ = 0;
    ELSE;
@
 DECIDE WHETHER TO ACCEPT CURRENT POINT
@
    V = RNDU(1,1);
    P = EXP(-DPHI/T);
    IREJ = 1;
      IF V < P;
       IREJ = 0;
       BX = XS;
       PHI0 = PHI1;
       ENDIF;
    ENDIF;
```

```
@
 UPDATE SCUM;
@
 SCUM[ITER,1] = PHI0;
@  ITER;; IREJ;; PHI0;; PHIOPT;@
@
 DECIDE WHETHER TO STOP
@
 STP = 1;
 IF ITER > 10;
  ITMP = ITER - 10;
  PBAR = MEANC(SCUM[ITMP:ITER,1]);
  STP = ABS((PBAR - PHIOPT)/PBAR);
  ENDIF;
 IF ITER > 10 AND STP < PSTOP;
  "STP = ";; STP;
  GOTO SP;
  ENDIF;
 ITER = ITER + 1;
 ENDO;
 SP:
@
 NOW DO NEWTON-RAPHSON
@
 ITER = 1;
 BX = XOPT;
 DO UNTIL ITER > MAXINR;
@
  COMPUTE THE THE OBJECTIVE FUNCTION AND ITS DERIVATIVES
@
 ITER;
 ARG = XVAR[.,1:K]*BX + XVAR[.,KPL];
 PHI0 = SY.*KFCN(ARG/SIG,N,H);
 PHI0 = MEANC(PHI0);
 T = SY.*XVAR[.,1:K].*DKFCN(ARG/SIG,N,H)./SIG;
 TS = MEANC(T);
 Q = SY.*XVAR[.,1:K].*DDKFCN(ARG/SIG,N,H)./(SIG^2);
 Q = XVAR[.,1:K]'Q/N;
 IMETH = 0;
 ISING = 0;
 TRAP 1;
 QINV = -INVPD(-Q);
@
 TEST WHETHER Q IS POSITIVE DEFINITE
@
   IF SCALERR(QINV) > 0;
    EIGVL = EIGH(Q);
    SGN = SUMC((EIGVL .< 0));
      IF SGN > 0.5 AND IRET <= 10;
       IRET = IRET + 1;
       GOTO ANNL;
       ELSEIF SGN > 0.5 AND IRET > 10;
       DIR = TS;
       IMETH = 1;
```

```
        GOTO SRCH;
        ENDIF;
    RNK = RANK(Q);
      IF RNK > 0;
        QINV = PINV(Q);
        ELSE;
        DIR = TS;
        IMETH = 1;
        GOTO SRCH;
      ENDIF;
    ELSE;
    DIR = -QINV*TS;
ENDIF;
TEST = TS'*DIR;
SRCH:
TEST = TS'DIR;
IF TEST < 0;
   IF IRET <= 10;
   IRET = IRET + 1;
   GOTO ANNL;
   ELSE;
   DIR = TS;
   ENDIF;
ENDIF;
STEPP = 1;
IT = 1;
DO UNTIL IT > 13;
  XS = BX + STEPP*DIR;
  ARG = XVAR[.,1:K]*XS + XVAR[.,KPL];
  PHI1 = SY.*KFCN(ARG/SIG,N,H);
  PHI1 = MEANC(PHI1);
  DPHI = PHI1 - PHI0;
    IF PHI1 > PHI0;
    GOTO STP;
    ENDIF;
  STEPP = 0.5*STEPP;
  IT = IT + 1;
 ENDO;
STP:
IF IT > 14;
  GOTO CONV;
  ENDIF;
PHI0 = PHI1;
TESTX = ABS(BX - XS)./(ABS(BX) + 1.E-7);
TESTX = SQRT(TESTX'TESTX);
BX = XS;
TESTG = SQRT(TS'TS);
IMETH;; IT;; DPHI;; TEST;
IF TESTG < PSTOP AND TESTX < PSTOP;
  FAIL = 0;
GOTO CONV;
  ENDIF;
  ITER = ITER + 1;
  ENDO;
```

```
@
 COMPUTE STANDARD ERRORS, BIAS CORRECTION AND ESTIMATED OPTIMAL BANDWIDTH
@
 CONV:
 OUTPUT ON;
 "OPTIMAL VALUES: ";
  XOPT';; PHIOPT;
@
 COMPUTE STANDARD ERRORS, BIAS CORRECTION AND ESTIMATED OPTIMAL BANDWIDTH
@
 BX = XOPT;
 ARG = XVAR[.,1:K]*BX + XVAR[.,KPL];
 T = SY.*XVAR[.,1:K].*DKFCN(ARG/SIG,N,H)./SIG;
 QV = SY.*XVAR[.,1:K].*DDKFCN(ARG/SIG,N,H)./(SIG^2);
 QV = XVAR[.,1:K]'QV/N;
@
 QV = ZEROS(K,K);
 I = 1;
 DO UNTIL I > K;
   J = I;
   DO UNTIL J > K;
   TMP = SY.*XVAR[.,I].*XVAR[.,J].*DDKFCN(ARG/SIG,N,H)/(SIG^2);
   QV[I,J] = MEANC(TMP);
   QV[J,I] = QV[I,J];
   J = J + 1;
   ENDO;
   I = I + 1;
   ENDO;
@
 "";
 "THE QV MATRIX FOLLOWS:";
 "";
 QV;
 "";
 DHAT = (SIG/N)*T'T;
 "THE DHAT MATRIX FOLLOWS";
 "";
 DHAT;
 "";
   SDEL = SIG^DEL;
   TDEL = MEANC(SY.*XVAR[.,1:K].*DKFCN(ARG/SDEL,N,H))/SDEL;
   AAHAT = (1/(SDEL^H))*TDEL;
 "THE A VECTOR FOLLOWS";
 "";
 AAHAT';
 "";
@
   SMALL-SAMPLE CORRECTION FOR AHAT AND DHAT
@
   ACORR = 1 - (SIG/SDEL)^H;
   AHAT = AAHAT/ACORR;
   TMP = SIG^H;
@
 BIAS CORRECTION AND STANDARD ERRORS
```

```
@
   ABIAS = -TMP*INV(QV)*AAHAT;
@
   "";
   "ESTIMATED COVARIANCE MATRIX FOLLOWS";
   "";
@
   AVAR = INV(QV)*DHAT*INV(QV)/(N*SIG);
@
   AVAR;
   "";
@
   XCORR = BX - ABIAS;
   STD = SQRT(DIAG(AVAR));
@
 OPTIMAL BANDWIDTH
@
 TOP = SUMC(DIAG(INV(QV)*DHAT*INV(QV)));
 BOT = 2*H*SUMC(DIAG(AHAT'*INV(QV)*INV(QV)*AHAT));
 LAM = TOP/BOT;
 GMA = 1/(2*H + 1);
 SIG0 = (LAM/N)^GMA;
 "BIAS CORRECTION: ";; ABIAS';
 "";
 "";
 "UNCORRECTED PARAMETERS, CORRECTED PARAMETERS, AND STD. ERRORS:";
 "COST DIFFERENCE IN DOLLARS HAS COEFFICIENT OF 1 BY NOMALIZATION;"
 "";
 "PARAMETER NAMES:"
 "";
 "    ";; $PARNAME';
 "";
 "         ";;BX~XCORR~STD;
 "";
 "OPTIMAL SIGMA: ";; SIG0;
@
 RCOUNT = RCOUNT/RSUM;
 "PROBABILITY OF REJECTING AN UNFAVORABLE STEP:  ";; RCOUNT;
@
 OUTPUT OFF;
 SIG = SIG + 0.05;
 RPT = RPT + 1;
 ENDO;
 RUNTM = ETHSEC(TSTRT,DATE)/360000;
 "";
 "RUNNING TIME IN HOURS:  ";; RUNTM;
 OUTPUT OFF;
 CLEAR YZ,SY,YVAR,XVAR,ARG;

/*************************************************************/
/********* PROCEDURES ****************************************/
/*************************************************************/

PROC KFCN(U,N,H);
```

```
@
 PROCEDURE TO EVALUATE THE SMOOTHING FUNCTION OF SMOOTHED MAXIMUM
 SCORE ESTIMATION.
@
 LOCAL K,A;

 IF H == 2;
   K = CDFN(U);  @ CUMULATIVE NORMAL DISTRIBUTION@

@
 K  BASED ON 2'ND ORDER QUADRATIC KERNEL WITH SUPPORT (-A,A)

 A = 2.5;
 U = U/A;
 K = ZEROS(N,1) + (0.5 + (15/16)*(U - (2/3)*(U^3) + (U^5)/5)).*(U .> -1)
       .*(U .< 1) + ONES(N,1).*(U .>= 1);
 KP = (1/A)*(15/16)*(ONES(N,1) - 2*(U^2) + U^4);
@
   ELSEIF H == 4;
@
 K BASED ON 4'TH ORDER QUARTIC KERNEL WITH SUPPORT (-A,A)
@
 A = 5;
 U = U/A;
 K = ZEROS(N,1) + (0.5 + (105/64)*(U - (5/3)*(U^3) + (7/5)*(U^5)
     - (3/7)*(U^7))).*(U .> -1).*(U .< 1) + ONES(N,1).*(U .>= 1);
 ENDIF;
 RETP(K);
 CLEAR K;
 ENDP;


/*******************************************************/

PROC DKFCN(U,N,H);
@
 PROCEDURE TO EVALUATE THE FIRST DERIVATIVE OF THE SMOOTHING FUNCTION
 OF SMOOTHED MAXIMUM SCORE ESTIMATION.
@
 LOCAL KFP,A;

 IF H == 2;
   KFP = PDFN(U);  @ CUMULATIVE NORMAL DISTRIBUTION@
@
 K  BASED ON 2'ND ORDER QUADRATIC KERNEL WITH SUPPORT (-A,A)

 A = 2.5;
 U = U/A;
 KFP = (1/A)*(15/16)*(ONES(N,1) - 2*(U^2) + U^4);
 KFP = KFP.*(U .> -1).*(U .< 1);
@
 ELSEIF H == 4;
@
 K BASED ON 4'TH ORDER QUARTIC KERNEL WITH SUPPORT (-A,A)
```

```
@
  A = 5;
  U = U/A;
  KFP = (1/A)*(105/64)*(ONES(N,1) - 5*(U^2) + 7*(U^4) - 3*(U^6))
        .*(U .> -1).*(U .< 1);
  ENDIF;
  RETP(KFP);
  CLEAR KFP;
  ENDP;

/*********************************************************/

PROC DDKFCN(U,N,H);
@
  PROCEDURE TO EVALUATE THE SECOND DERIVATIVE OF THE SMOOTHING FUNCTION
  OF SMOOTHED MAXIMUM SCORE ESTIMATION.
@
  LOCAL KFPP,A;

  IF H == 2;
    KFPP = -U.*PDFN(U);  @ CUMULATIVE NORMAL DISTRIBUTION @

@
  K  BASED ON 2'ND ORDER QUADRATIC KERNEL WITH SUPPORT (-A,A)

  A = 2.5;
  U = U/A;
  KFPP = (1/A^2)*(15/4)*(-U + U^3);
  KFPP = KFPP.*(U .> -1).*(U .< 1);
@
    ELSEIF H == 4;
@
  K BASED ON 4'TH ORDER QUARTIC KERNEL WITH SUPPORT (-A,A)
@
    A = 5;
    U = U/A;
    KFPP = (1/A^2)*(105/64)*(-10*U + 28*(U^3) - 18*(U^5)).*(U .> -1).*(U .< 1);
  ENDIF;
  RETP(KFPP);
  CLEAR KFPP;
  ENDP;
```