```
C    ===============================================================
c       Sensitivity analysis for
C       EXAMPLE 4: a function of non-Gaussian variables (Lognormally distributed)
C       METHOD: PDD-SPA option 1
c           SENSITIVITY CODE
C
C
C
C       N = 6
C       Y = X1 + 2*X2 + 2*X_3 + X4 - 5*X5 - 5*X6
c
C     X(I) are lognormally distributed,  mu1 ... mu4 = 120, mu5 = 50, mu6 = 40, COV = c \in [0.1, 0.7]    STIELTJES PROCEDURE
C     GAUSS QUADRATURE RULE BY STIELTJES PROCEDURE
c
C     SUBROUTINES:
c           PDD_SEN_PF_SPA      :       GET SENSITIVITY BY PDD-SPA METHOD
C           TS_SPA              :       CALCULATE SADDLEPOINT
C           CDF_SPA             :       CALCULATE CDF VALUE AT A SPECIFIC POINT
C           SEN_CDF_SPA         :       CALCULATE SENSITIVITIES OF CDF AT A SPECIFIC POINT
C           DIRECT_MCS          :       DIRECT MONTE CARLO SIMULATION TO GENERATE BENCHMARK RESULTS

C
C***********************************************************************************
c     Copyright (c) 2010-2014 [Xuchun Ren], All Rights Reserved.

      SUBROUTINE PDD_SEN_PF_SPA (XMU_IN,XMU_IN2, cov, NSAMP,NBAS,
     1   NBAS_SM, NGAUSS,
     1 NGAUSS_SM, IID,  NGAUSS_GQ, IFLAG_Y, IFLAG_MCS, TS_OUT, PF,
     2     PF_SEN, PF_MCS, PF_SEN_MCS,
     3   IFLAG_AIJK, NVAR, ncum, ng_cum, ng_cum_sen, ntheta)
c                          set ng_cum == ng_cum_sen
        USE FAC_INT
        USE RNNOF_INT
        USE RNSET_INT
        USE GQRUL_INT
        USE UMACH_INT
        USE GAMMA_INT

        USE DFLIB
        INCLUDE "link_fnl_shared.h"
CC     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR EXAMPLE 4.1 OF PAPER "ROBUST DESIGN WITH ARBITRARY DISTRIBUTIONS USING GAUSS-TYPE QUADRATURE FORMULA" BY S.H. LEE, W. CHEN, ETC.
C       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        IMPLICIT REAL*8 (A-H,O-Z)
        DOUBLE PRECISION XMU_IN(200),XMU_IN2(200), cov, XSIG_IN(200),
        1      MOMSEN_OUT(10,2)
        REAL*8 IND
        REAL*8 X(20),XMU(20),SIG(20),XX(50),QX(50),QW(50),QXFIX(2),
        1          NMU(20), NSIG(20), SYUI(20), TMPA1(20), TMPA2(20),
     2          SMU_A(200),SSIG_A(200),SMU_A_TMP(200),SSIG_A_TMP(200)
      REAL*8 XX_SM(50),QX_SM(50),QW_SM(50),QXFIX_SM(2),X1(20)
        REAL*8 YR1(:,:),YR2(:,:,:,:),YR3(:,:,:,:,:,:),H(:,:,:)
        REAL*8 SMR1(:,:),SMR2(:,:,:,:),SMR3(:,:,:,:,:,:),SMH(:,:,:),
     1      SGR1(:,:),SGR2(:,:,:,:),SGR3(:,:,:,:,:,:),SGH(:,:,:)
C       REAL*8 ALFA(:,:),PSIF(:,:),BITA(:,:,:,:),GAMA(:,:,:,:,:,:)
        REAL*8 ALFA(10,10),PSIF(:,:),BITA(:,:,:,:),GAMA(:,:,:,:,:,:)
        REAL*8 ALFA_SM(:,:),PSIF_SM(:,:),BITA_SM(:,:,:,:),
        1          GAMA_SM(:,:,:,:,:,:),
     2      ALFA_SG(:,:),PSIF_SG(:,:),BITA_SG(:,:,:,:),
        1          GAMA_SG(:,:,:,:,:,:), PSIF_N(:,:),
     3      ALFA_SMIID(:,:),ALFA_SGIID(:,:),SCMU(:,:,:),SCSIG(:,:,:),
     4      YPDD(:,:,:), H_GQ(:,:,:), Y0
      REAL*8 QXN(20,50),QWN(20,50),XXN(20,50),RDD_MU(20), XXN_SM(20,50),
     1      QWN_SM(20,50),QXN_SM(20,50)
      INTEGER NGAUSS(20), NBAS(20), NGAUSS_MAX, NBAS_MAX, NGAUSS_SM(20),
     1      NBAS_SM(20), NGAUSS_SM_MAX, NBAS_SM_MAX, I_TEMP

C      ALLOCATABLE YR1,YR2,YR3,H,ALFA,PSIF,BITA,GAMA,SMR1,SMR2,SMR3,SMH,
      ALLOCATABLE YR1,YR2,YR3,H,PSIF,BITA,GAMA,SMR1,SMR2,SMR3,SMH,
     1              ALFA_SM,PSIF_SM,BITA_SM,GAMA_SM,
     2      SGR1,SGR2,SGR3,SGH,PSIF_N,
     3              ALFA_SG,PSIF_SG,BITA_SG,GAMA_SG,
     4          ALFA_SGIID, ALFA_SMIID,SCMU,SCSIG,YPDD,H_GQ

        REAL*8 A(20,20),B(20,20),AIJK(10,10,10,10,10,10),
        1      AIJKC(10,10,10,10,10,10,10)
        INTEGER NBAS_A(20), NO_GAUSS, ncum, ng_cum, ng_cum_sen, ii(N),ir(N),
     1  ntheta, IW1,np
c     ntheta is the # of distribution parameters of each variable
      REAL*8 AF1(10,10), BT1(10,10,10,10), GM1(10,10,10,10,10,10),
     1  ALFA_OUT(10,10), X_TEMP(20),AIJK_TEMP(10,10,10,10,10,10)

      REAL*8 XGUESS1,TS_OUT, Y_LIM, PF, PF_SEN(10,10), PF_MCS,
     1  PF_SEN_MCS(20,10), YU(10,10),WU(10,10),aa(20), bb(20),
     2  qqw(50,50), qqx(50,50),qqxn(10,50,50),qqwn(10,50,50),
     3  aan(10,20), bbn(10,20),wg, xg(20), cum(20), cum_sen(20,20,10),
     4  yp(20),sc(20,10), ts(20), ylim, cdf_sen(20,20,10), cdf(20),
     5  cdf_sen_mcs(20,10),cdf_mcs,aan1(10,20), bbn1(10,20),
     6  cumk(20), cumk_sen(20,20,10), Ky(20), pf_pmcs,
     7  pf_sen_pmcs(20,10)
c       cum_sen(ncum,N,ntheta) , cdf_sen(ncum,N, ntheta)

C      INTEGER N
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N
        COMMON /XR01/ N_F, N_GF, N_GC, N_EF,N_EC, N_EF1, N_EC1, N_EF2, N_EC2
      COMMON /XR02/ AIJKC
      COMMON /XR06/ NO_GAUSS
      COMMON /XR07/ IW1

      COMMON /SPA01/ Y0, YU,WU, NBAS_SPA, NGAUSS_SPA
      COMMON /SPA02/ AF1, BT1, GM1

        CHARACTER(80) MSG,MSG1
        MSG = 'PROGRAM NSAMP NBAS NBAS_SM NGAUSS NGAUSS_SM FN IID NYQ
     1         NGAUSS_GQ'
        MSG1 = 'IID = 0(INDEPENDENT), 1(IID), 2(DEPENDNENT) '

C

       do i = 1,N
       RDD_MU(i) = XMU_IN2(i)
       enddo

       ylim = 0.
       NO_GAUSS = 0

        DO I =1,20
          X1(I)=0.
        ENDDO
      NBAS_A = 0.
      NBAS_A(1:9) = NBAS(1:9)
C       N = 2
        PI = 4.*DATAN(1.0D0)
        DO I = 1,N
        XMU(I) = XMU_IN2(I)
        SIG(I) = XMU_IN2(I)*cov
        ENDDO

      DO I = 1,N
            NMU(I) = XMU_IN2(I)
              NSIG(I) = XMU_IN2(I)*cov
              XSIG_IN(I) = NSIG(I)
        ENDDO

        NGAUSS_MAX=MAXVAL(NGAUSS)
        NBAS_MAX=MAXVAL(NBAS)
        NGAUSS_SM_MAX=MAXVAL(NGAUSS_SM)
        NBAS_SM_MAX=MAXVAL(NBAS_SM)

      I_TEMP = 1
      DO I = 1, N
        I_TEMP = I_TEMP*NGAUSS(I)* NGAUSS_SM(I)
        IF (I_TEMP .NE. 0) I_TEMP = 1
      ENDDO

        IF (I_TEMP .EQ. 0) THEN
         PRINT*, "NGAUSS =0 OR NGAUSS_SM = 0"
         STOP
      ENDIF

      qqxn = 0.
      qqwn = 0.

c       CALL DGQRUL (NGAUSS(1),4,0.,0.,0,QXFIX,QX,QW)
c       CALL DGQRUL (NGAUSS_SM(1),4,0.,0.,0,QXFIX_SM,QX_SM,QW_SM)
      aan1 = 0.
      bbn1 = 0.
      i_flag = 0
        DO J = 1, N
      qqx = 0.
      qqw = 0.
        call DGQRUL_LOGN2(XMU_IN(J), XSIG_IN(J), i_flag, NBAS(1),
     1  aa, bb, NGAUSS(1), qqx, qqw )
      aan1(J,:) = aa(:)
      bbn1(J,:) = bb(:)
c      qqxn(J,:,:) = qqx(:,:)
c      qqwn(J,:,:) = qqw(:,:)
      QXN(J,:) = qqx(NGAUSS(1),:)
      QWN(J,:) = qqw(NGAUSS(1),:)
      XXN(J,:) = qqx(NGAUSS(1),:)
C--------------  no QXN_SM  since we using numerical quadrature  ---------
        ENDDO

        ALLOCATE( YR1(N,NGAUSS_MAX),YR2(N,N,NGAUSS_MAX,NGAUSS_MAX),
        1  H(N,NBAS_MAX,NGAUSS_MAX),
C     &          ALFA(N,NBAS_MAX),PSIF(N,NBAS_MAX),PSIF_N(N,NBAS_MAX),
     &          PSIF(N,NBAS_MAX),PSIF_N(N,NBAS_MAX),
     1      BITA(N,N,NBAS_MAX,NBAS_MAX),
     &          YR3(N,N,N,NGAUSS_MAX,NGAUSS_MAX,NGAUSS_MAX),
     &          GAMA(N,N,N,NBAS_MAX,NBAS_MAX,NBAS_MAX),
     &   SMR1(N,NGAUSS_SM_MAX),SMR2(N,N,NGAUSS_SM_MAX,NGAUSS_SM_MAX),
     &   SMH(N,NBAS_SM_MAX,NGAUSS_SM_MAX),ALFA_SM(N,NBAS_SM_MAX),
     &   PSIF_SM(N,NBAS_SM_MAX),
     &   BITA_SM(N,N,NBAS_SM_MAX,NBAS_SM_MAX),
     &          SMR3(N,N,N,NGAUSS_SM_MAX,NGAUSS_SM_MAX,NGAUSS_SM_MAX),
     &          GAMA_SM(N,N,N,NBAS_SM_MAX,NBAS_SM_MAX,NBAS_SM_MAX),
     &   SGR1(N,NGAUSS_SM_MAX),SGR2(N,N,NGAUSS_SM_MAX,NGAUSS_SM_MAX),
     &   SGH(N,NBAS_SM_MAX,NGAUSS_SM_MAX),ALFA_SG(N,NBAS_SM_MAX),
     &   PSIF_SG(N,NBAS_SM_MAX),
     &   BITA_SG(N,N,NBAS_SM_MAX,NBAS_SM_MAX),
     &          SGR3(N,N,N,NGAUSS_SM_MAX,NGAUSS_SM_MAX,NGAUSS_SM_MAX),
     &          GAMA_SG(N,N,N,NBAS_SM_MAX,NBAS_SM_MAX,NBAS_SM_MAX),
     &    ALFA_SGIID(N,NBAS_SM_MAX),ALFA_SMIID(N,NBAS_SM_MAX) )

      ALLOCATE(H_GQ(N,NBAS_MAX,NGAUSS_GQ),SCMU(NGAUSS_GQ,
     1  NGAUSS_GQ,NGAUSS_GQ),SCSIG(NGAUSS_GQ,NGAUSS_GQ,NGAUSS_GQ),
     1          YPDD(NGAUSS_GQ,NGAUSS_GQ,NGAUSS_GQ))

C
C       INITIALIZE COEFFICIENTS
C

        Y0 = 0.

      IF (IID .EQ. 1) THEN
C         IF (IID .EQ. 1 .AND. (NGAUSS .NE. NGAUSS_SM .OR.
C     1           NBAS .NE. NBAS_SM)) THEN
C         PRINT*,' NGAUSS != NGAUSS_SM  or NBAS != NBAS_SM '
C         STOP
C         ENDIF
      ENDIF

        DO I = 1,N
         DO J = 1,NBAS(I)
         ALFA(I,J) = 0.
         ENDDO
        ENDDO

        DO I1 = 1,N
         DO I2 = 1,N
          DO J1 = 1,NBAS(I1)
           DO J2 = 1,NBAS(I2)
           BITA(I1,I2,J1,J2) = 0.
           ENDDO
          ENDDO
         ENDDO
        ENDDO

        DO I1 = 1,N
         DO I2 = 1,N
          DO I3 = 1,N
           DO J1 = 1,NBAS(I1)
            DO J2 = 1,NBAS(I2)
             DO J3 = 1,NBAS(I3)
             GAMA(I1,I2,I3,J1,J2,J3) = 0.
             ENDDO
            ENDDO
           ENDDO
          ENDDO
         ENDDO
        ENDDO

        ALFA = 0.
        BITA = 0.
        GAMA = 0.

C       CALCULATE/STORE RESPONSE AT MEAN INPUT (YM  SSM)
C
        DO I = 1,N
        X(I) = RDD_MU(I)
        ENDDO
        IF (IFLAG_AIJK .EQ. 0)  THEN
        CALL RESPON (X,YM, IFLAG_Y)
C      CALL RESPON (X,YM, IFLAG_Y)
        NO_GAUSS = NO_GAUSS + 1
        IF (IFLAG_Y .EQ. 1) N_EF1 = N_EF1 +1
        IF (IFLAG_Y .EQ. 2) N_EC1 = N_EC1 +1
        ELSEIF(IFLAG_AIJK .EQ. 1) THEN
        CALL RESPON_AIJK(NBAS_A,AIJKC(IFLAG_Y,:,:,:,:,:,:), X, YM)
        ELSE
        PRINT*, 'WRONG IFLAG = ', IFLAG
        STOP
        ENDIF

C       CALL SCOR (X,SMM,SGM)

C
C       CALCULATE/STORE UNIVARIATE RESPONSES AT GAUSS POINTS (YR1)
C
      YR1 = 0.
        DO IX = 1,N
         DO I = 1,N
         X(I) = RDD_MU(I)
         ENDDO

          DO IGAUSS = 1,NGAUSS(IX)
          X(IX) = XXN(IX,IGAUSS)
            IF (IFLAG_AIJK .EQ. 0)  THEN
            CALL RESPON (X,Y, IFLAG_Y)
C          CALL RESPON (X,Y, IFLAG_Y)

            NO_GAUSS = NO_GAUSS + 1
          IF (IFLAG_Y .EQ. 1) N_EF1 = N_EF1 +1
            IF (IFLAG_Y .EQ. 2) N_EC1 = N_EC1 +1

            ELSEIF(IFLAG_AIJK .EQ. 1) THEN
            CALL RESPON_AIJK(NBAS_A,AIJKC(IFLAG_Y,:,:,:,:,:,:), X, Y)

            ELSE
            PRINT*, 'WRONG IFLAG_AIJK = ', IFLAG_AIJK
            STOP
            ENDIF

          YR1(IX,IGAUSS) = Y
          ENDDO
        ENDDO

C
C       CALCULATE/STORE BIVARIATE RESPONSES AT GAUSS POINTS (YR2)
C
      YR2 = 0.
      IF (NVAR .GE. 2) THEN
        DO IX = 1,N-1
         DO JX = IX+1,N
          DO I = 1,N
          X(I) = RDD_MU(I)
          ENDDO

C                HERE CALCULATE THE VALUE OF Y AT THE CASE ONLY VARIABLE X(IX) AND X(JX) NONZERO  IN NGAUSS X NGAUSS INTEGRAL POINTS
           DO IGAUSS = 1,NGAUSS(IX)
            DO JGAUSS = 1,NGAUSS(JX)
            X(IX) = XXN(IX,IGAUSS)
            X(JX) = XXN(JX,JGAUSS)
            IF (IFLAG_AIJK .EQ. 0)  THEN
            CALL RESPON (X,Y, IFLAG_Y)
C          CALL RESPON (X,Y, IFLAG_Y)
            NO_GAUSS = NO_GAUSS + 1
            IF (IFLAG_Y .EQ. 1) N_EF2 = N_EF2 +1
            IF (IFLAG_Y .EQ. 2) N_EC2 = N_EC2 +1
            ELSEIF(IFLAG_AIJK .EQ. 1) THEN
            CALL RESPON_AIJK(NBAS_A,AIJKC(IFLAG_Y,:,:,:,:,:,:), X, Y)
            ELSE
            PRINT*, 'WRONG IFLAG_AIJK = ', IFLAG_AIJK
            STOP
            ENDIF

            YR2(IX,JX,IGAUSS,JGAUSS) = Y

            ENDDO
           ENDDO
         ENDDO
        ENDDO
        ENDIF

C
C
C       CALCULATE/STORE TRIVARIATE RESPONSES AT GAUSS POINTS (YR3)
C
      YR3 = 0.
      IF (NVAR .EQ. 3) THEN
        DO IX = 1,N-2
         DO JX = IX+1,N-1
          DO KX = JX+1,N
           DO I = 1,N
           X(I) = RDD_MU(I)
           ENDDO

            DO IGAUSS = 1,NGAUSS(IX)
             DO JGAUSS = 1,NGAUSS(JX)
              DO KGAUSS = 1,NGAUSS(KX)
              X(IX) = XXN(IX,IGAUSS)
              X(JX) = XXN(JX,JGAUSS)
              X(KX) = XXN(KX,KGAUSS)
                IF (IFLAG_AIJK .EQ. 0)  THEN
                CALL RESPON (X,Y, IFLAG_Y)
C              CALL RESPON (X,Y, IFLAG_Y)
                NO_GAUSS = NO_GAUSS + 1
                IF (IFLAG_Y .EQ. 1) N_EF2 = N_EF2 +1
                IF (IFLAG_Y .EQ. 2) N_EC2 = N_EC2 +1
                ELSEIF(IFLAG_AIJK .EQ. 1) THEN
                CALL RESPON_AIJK(NBAS_A,AIJKC(IFLAG_Y,:,:,:,:,:,:), X, Y)
                ELSE
                PRINT*, 'WRONG IFLAG_AIJK = ', IFLAG_AIJK
                STOP
                ENDIF
              YR3(IX,JX,KX,IGAUSS,JGAUSS,KGAUSS) = Y
              ENDDO
             ENDDO
            ENDDO

          ENDDO
         ENDDO
        ENDDO
        ENDIF

C
C       CALCULATE/STORE BASIS FUNCTIONS AT GAUSS POINTS (H)
C     REMARK:  HERE THE DO J SHOULD BE PUT INSIDE AFTER CALL BASIS TO REDUCE COMPUTATION COST
        DO I = 1,N
         DO J = 1,NBAS(I)
          DO L = 1,N
          X(L) = RDD_MU(L)
          ENDDO
           DO IGAUSS = 1,NGAUSS(I)
           X(I) = XXN(I,IGAUSS)
         CALL BASIS2(X,PSIF,N,NBAS(I),aan1,bbn1)
         H(I,J,IGAUSS) = PSIF(I,J)
           ENDDO
         ENDDO
        ENDDO

C
C       CALCULATE COEFFICIENT Y0
C

        YU0 = 0.
        DO IX = 1,N
         DO IGAUSS = 1,NGAUSS(IX)
         YU0 = YU0 + YR1(IX,IGAUSS)*QWN(IX,IGAUSS)
       ENDDO
        ENDDO

C
        YB = 0.
        DO IX = 1,N-1
         DO JX = IX+1,N
          DO IGAUSS = 1,NGAUSS(IX)
           DO JGAUSS = 1,NGAUSS(JX)
           YB = YB + YR2(IX,JX,IGAUSS,JGAUSS)*QWN(IX,IGAUSS)*
     &        QWN(JX,JGAUSS)
         ENDDO
          ENDDO
         ENDDO
        ENDDO

C
        YT = 0.
        DO IX = 1,N-2
         DO JX = IX+1,N-1
          DO KX = JX+1,N
           DO IGAUSS = 1,NGAUSS(IX)
            DO JGAUSS = 1,NGAUSS(JX)
             DO KGAUSS = 1,NGAUSS(KX)
             YT = YT + YR3(IX,JX,KX,IGAUSS,JGAUSS,KGAUSS)*
     &          QWN(IX,IGAUSS)*QWN(JX,JGAUSS)*
     &          QWN(KX,KGAUSS)
           ENDDO
            ENDDO
           ENDDO
          ENDDO
         ENDDO
        ENDDO

C
C
      NT = N
      IF (NVAR .EQ. 3) THEN
      Y0 = YT - DFLOAT(NT-3)*YB + DFLOAT((NT-2)*(NT-3))*YU0/2.
     &     - DFLOAT((NT-1)*(NT-2)*(NT-3))*YM/6.

      ELSE IF (NVAR .EQ. 2) THEN
      Y0 =  YB
     &     - DFLOAT((NT-2))*YU0 + DFLOAT((NT-2)*(NT-1))*YM/2.
      ELSE IF (NVAR .EQ. 1) THEN
      Y0 =  YU0
     &     - DFLOAT((NT-1))*YM
      ENDIF

c     for 2 variable, univariate expansion
C      Y0 =  DFLOAT((N-1))*YU0
C    &     - DFLOAT((N-1))*YM

C           Y0 =  YU0
C     &     - DFLOAT((NT-1))*YM
C
C           Y0 =  YB
C     &     - DFLOAT((NT-2))*YU0 + DFLOAT((NT-2)*(NT-1))*YM/2.
C

C
C       CALCULATE COEFFICIENTS ALFA(I,J)
        DO I = 1,N
         DO J = 1,NBAS(I)
C
         YYU = 0.
          DO IGAUSS = 1,NGAUSS(I)
          YYU = YYU + YR1(I,IGAUSS)*H(I,J,IGAUSS)*QWN(I,IGAUSS)
          ENDDO

C      DO 10 IX = 1,N-1
C      DO 10 JX = IX+1,N
C      IF (IX .NE. I .AND. JX .NE. I) GO TO 10
C       DO IGAUSS = 1,NGAUSS(IX)
C        DO JGAUSS = 1,NGAUSS(JX)
C        IF (IX .EQ. I) KGAUSS = IGAUSS
C        IF (JX .EQ. I) KGAUSS = JGAUSS
C        YYB = YYB + YR2(IX,JX,IGAUSS,JGAUSS)*H(I,J,KGAUSS)*
C     &           (QW(IGAUSS)/DSQRT(PI))*(QW(JGAUSS)/DSQRT(PI))
C        ENDDO
C       ENDDO
C10    CONTINUE
C
         YYB = 0.
           DO 10 IX = 1,N-1
           DO 10 JX = IX+1,N
           IF (IX .NE. I .AND. JX .NE. I) GO TO 10
            DO IGAUSS = 1,NGAUSS(IX)
             DO JGAUSS = 1,NGAUSS(JX)
             IF (IX .EQ. I) KGAUSS = IGAUSS
             IF (JX .EQ. I) KGAUSS = JGAUSS
             YYB = YYB + YR2(IX,JX,IGAUSS,JGAUSS)*H(I,J,KGAUSS)*
     &           QWN(IX,IGAUSS)*QWN(JX,JGAUSS)
             ENDDO
            ENDDO
10         CONTINUE

C
C
          YYT = 0.
           DO 20 IX = 1,N-2
           DO 20 JX = IX+1,N-1
           DO 20 KX = JX+1,N
           IF (IX .NE. I .AND. JX .NE. I .AND. KX .NE. I) GO TO 20
            DO IGAUSS = 1,NGAUSS(IX)
             DO JGAUSS = 1,NGAUSS(JX)
              DO KGAUSS = 1,NGAUSS(KX)
              IF (IX .EQ. I) MGAUSS = IGAUSS
              IF (JX .EQ. I) MGAUSS = JGAUSS
              IF (KX .EQ. I) MGAUSS = KGAUSS
              YYT = YYT + YR3(IX,JX,KX,IGAUSS,JGAUSS,KGAUSS)*H(I,J,MGAUSS)*
     &           QWN(IX,IGAUSS)*QWN(JX,JGAUSS)*
     &           QWN(KX,KGAUSS)
                 ENDDO
             ENDDO
            ENDDO
20         CONTINUE
C
      NT = N
      IF (NVAR .EQ. 3) THEN
      ALFA(I,J) = YYT - DFLOAT(NT-3)*YYB + DFLOAT((NT-2)*(NT-3))*YYU/2.

      ELSE IF (NVAR .EQ. 2) THEN
      ALFA(I,J) =  YYB
     &     - DFLOAT((NT-2))*YYU
      ELSE IF (NVAR .EQ. 1) THEN
      ALFA(I,J) =  YYU
      ENDIF

c          ALFA(I,J) = YYT - DFLOAT(NT-3)*YYB + DFLOAT((NT-2)*(NT-3))*YYU/2.
C
         ENDDO
        ENDDO

C
C       CALCULATE COEFFICIENTS BITA(I1,I2,J1,J2)
C
        DO I1 = 1,N-1
         DO I2 = I1+1,N
          DO J1 = 1,NBAS(I1)
           DO J2 = 1,NBAS(I2)
C
          YYYB = 0.
            DO IGAUSS = 1,NGAUSS(I1)
             DO JGAUSS = 1,NGAUSS(I2)
             YYYB = YYYB +
     &     YR2(I1,I2,IGAUSS,JGAUSS)*H(I1,J1,IGAUSS)*H(I2,J2,JGAUSS)*
     &     QWN(I1,IGAUSS)*QWN(I2,JGAUSS)
             ENDDO
            ENDDO
C
          YYYT = 0.
           DO 30 IIX = 1,N-2
           DO 30 JJX = IIX+1,N-1
           DO 30 KKX = JJX+1,N
           IF (IIX .NE. I1 .AND. IIX .NE. I2 .AND.
     &       JJX .NE. I1 .AND. JJX .NE. I2 .AND.
     &       KKX .NE. I1 .AND. KKX .NE. I2) GO TO 30
           IF (IIX .EQ. I1 .AND. (JJX .EQ. I2 .OR. KKX .EQ. I2)) GO TO 31
           IF (JJX .EQ. I1 .AND. (IIX .EQ. I2 .OR. KKX .EQ. I2)) GO TO 31
           IF (KKX .EQ. I1 .AND. (IIX .EQ. I2 .OR. JJX .EQ. I2)) GO TO 31
           GO TO 30
31        DO IGAUSS = 1,NGAUSS(IIX)
             DO JGAUSS = 1,NGAUSS(JJX)
              DO KGAUSS = 1,NGAUSS(KKX)
              IF (IIX .EQ. I1) MGAUSS1 = IGAUSS
              IF (JJX .EQ. I1) MGAUSS1 = JGAUSS
              IF (KKX .EQ. I1) MGAUSS1 = KGAUSS
              IF (IIX .EQ. I2) MGAUSS2 = IGAUSS
              IF (JJX .EQ. I2) MGAUSS2 = JGAUSS
              IF (KKX .EQ. I2) MGAUSS2 = KGAUSS
              YYYT = YYYT + YR3(IIX,JJX,KKX,IGAUSS,JGAUSS,KGAUSS)*
     &      H(I1,J1,MGAUSS1)*H(I2,J2,MGAUSS2)*QWN(IIX,IGAUSS)*
     &      QWN(JJX,JGAUSS)*QWN(KKX,KGAUSS)
                 ENDDO
             ENDDO
            ENDDO
30         CONTINUE
C
         NT = N
      IF (NVAR .EQ. 3) THEN
      BITA(I1,I2,J1,J2) = YYYT - DFLOAT(NT-3)*YYYB

      ELSE IF (NVAR .EQ. 2) THEN
      BITA(I1,I2,J1,J2) =  YYYB
      ELSE IF (NVAR .EQ. 1) THEN
      BITA(I1,I2,J1,J2) =  0.
      ENDIF
C          BITA(I1,I2,J1,J2) = YYYT - DFLOAT(NT-3)*YYYB

C       WRITE(*,*) I1, I2, J1, J2,   BITA(I1,I2,J1,J2)
C
           ENDDO
          ENDDO
         ENDDO
        ENDDO

C
C
C       CALCULATE COEFFICIENTS GAMA(I1,I2,I3,J1,J2,J3)
C
        DO I1 = 1,N-2
         DO I2 = I1+1,N-1
          DO I3 = I2+1,N
           DO J1 = 1,NBAS(I1)
            DO J2 = 1,NBAS(I2)
             DO J3 = 1,NBAS(I3)
C
             GAMA(I1,I2,I3,J1,J2,J3) = 0.
            DO IGAUSS = 1,NGAUSS(I1)
               DO JGAUSS = 1,NGAUSS(I2)
                DO KGAUSS = 1,NGAUSS(I3)
              GAMA(I1,I2,I3,J1,J2,J3) = GAMA(I1,I2,I3,J1,J2,J3) +
     &              YR3(I1,I2,I3,IGAUSS,JGAUSS,KGAUSS)*
     &        H(I1,J1,IGAUSS)*H(I2,J2,JGAUSS)*H(I3,J3,KGAUSS)*
     &        QWN(I1,IGAUSS)*
     &        QWN(I2,JGAUSS)*QWN(I3,KGAUSS)
                  ENDDO
               ENDDO
              ENDDO
C
             ENDDO
            ENDDO
           ENDDO
          ENDDO
         ENDDO
        ENDDO

c---------------------------  start to gauss quadrature to calculate up to ncum moments and their sensitivity  ---------------

      AF1 = 0.
      BT1 = 0.
      GM1 = 0.
      ALFA_OUT = 0.

      DO I = 1, N
      DO K = 1, NBAS_A(I)
      AF1(I,K) = ALFA(I,K)
      ALFA_OUT(I,K) = ALFA(I,K)
      ENDDO
      ENDDO

      DO I1 = 1, N-1
      DO I2 = I1+1, N
      DO K1 = 1, NBAS_A(I1)
      DO K2 = 1, NBAS_A(I2)
      BT1(I1,I2,K1,K2) = BITA(I1,I2,K1,K2)
      ENDDO
      ENDDO
      ENDDO
      ENDDO

      DO I1 = 1, N-2
      DO I2 = I1+1, N-1
      DO I3 = I2+1, N
      DO K1 = 1, NBAS_A(I1)
      DO K2 = 1, NBAS_A(I2)
      DO K3 = 1, NBAS_A(I3)
      GM1(I1,I2,I3,K1,K2,K3) = GAMA(I1,I2,I3,K1,K2,K3)
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO

      if (ng_cum_sen .lt. ng_cum .or. ng_cum .lt. NGAUSS(1)) then
        print*,'ERROR!: ng_cum_sen < ng_cum OR ng_cum  < NGAUSS(1)'
        stop
      endif

c     ****************  get gauss quadrature rule for moments ***************
      aan = 0.
      bbn = 0.
        DO J = 1, N
      qqx = 0.
      qqw = 0.
      aa = 0.
      bb = 0.
        call DGQRUL_LOGN2(XMU_IN(J), XSIG_IN(J), i_flag,  NBAS(1),
     1  aa, bb, ng_cum_sen + 1, qqx, qqw )
      aan(J,:) = aa(:)
      bbn(J,:) = bb(:)
      qqxn(J,:,:) = qqx(:,:)
      qqwn(J,:,:) = qqw(:,:)
      QXN(J,:) = qqx(NGAUSS(1),:)
      QWN(J,:) = qqw(NGAUSS(1),:)
      XXN(J,:) = qqx(NGAUSS(1),:)
        ENDDO

c     test for eval_PDD
       X_TEMP  = 0.0
       X_TEMP(1:6) = (/3,4,5,6,7,8/)
       CALL EVAL_PDD(aan,bbn,Y0, AF1, BT1, GM1, NBAS_A, N, X_TEMP,
     1   Y_PDD, NVAR)
C
C
      PRINT*, 'IFLAG_Y = ', IFLAG_Y
      PRINT*, 'Y_PDD = ', Y_PDD
      PRINT*, 'X_TEMP = ', X_TEMP(1:6)

c     ************  evaluate PDD at gauss points and calculate cumulants and their sensitivities  --------------
c     ---- central moments first  ----
      cum = 0.
      cum_sen = 0.
      do i = 1, ng_cum_sen**N
        do j = N, 1, -1
        if (j .eq. N) then
        ii(j) = mod(i-1,ng_cum_sen) + 1
        ir(j) = int((i-1)/ng_cum_sen)
        else
        ii(j) = mod(ir(j+1),ng_cum_sen) + 1
        ir(j) = int(ir(j+1)/ng_cum_sen)
        endif
        enddo
        wg = 1.0
        do j = 1,N
        xg(j) = qqxn(j,ng_cum_sen,ii(j))
        wg =  wg * qqwn(j,ng_cum_sen,ii(j))
        enddo
        call EVAL_PDD(aan,bbn,Y0, AF1, BT1, GM1, NBAS_A, N, xg ,y_pdd
     1 , NVAR)
        call scor_logn2(xg,sc)
        do j = 1,ncum
            yp(j) =  y_pdd**j
            cum(j) = cum(j) + yp(j)*wg
            do k1 = 1, N
            do k2 = 1, ntheta
            cum_sen(j,k1,k2) = cum_sen(j,k1,k2) + yp(j)*sc(k1,k2)*wg
            enddo
            enddo
        enddo
      enddo

c     ---- from central moements to cumulants including sensitivities ----
c     cumk(20), cumk_sen(20,20,10)
      do i = 1, ncum
            cumk(i) = cum(i)
            do k1 = 1, N
            do k2 = 1, ntheta
            cumk_sen(i,k1,k2) = cum_sen(i,k1,k2)
            enddo
            enddo
        if (i .gt. 1) then
          do j = 1,i-1
            cumk(i) = cumk(i) - DFAC(i-1)/(DFAC(j-1)*DFAC(i-j))*cumk(j)*
     1          cum(i-j)
            do k1 = 1, N
            do k2 = 1, ntheta
            cumk_sen(i,k1,k2) = cumk_sen(i,k1,k2) - DFAC(i-1)/
     1       (DFAC(j-1)*DFAC(i-j))*(cumk_sen(j,k1,k2)*cum(i-j)
     2       +cumk(j)*cum_sen(i-j,k1,k2))
            enddo
            enddo
          enddo
        endif
      enddo

c     **************** get all saddle points by 2nd, 4th, 6th .... cumulants *******
      ts = 0.
      call ts_spa(ncum, cumk, ylim, ts)

c     **************** calculate cdf *******
      call cdf_spa(ncum, cumk, ylim, ts, cdf, Ky)
      call sen_cdf_spa(N,ncum,ntheta, cumk, cumk_sen, ylim, ts, cdf_sen)

      call DIRECT_MCS(NSAMP, ylim, aan,bbn,Y0, AF1, BT1, GM1,
     1 NBAS_A,NVAR,
     1  PF_MCS, PF_SEN_MCS, pf_pmcs, pf_sen_pmcs)

      IF (IW1 .EQ. 0) THEN
          IW1 = 1
          OPEN(90,FILE='Ky.DAT',STATUS='UNKNOWN' )

          OPEN(100,FILE='TS.DAT',STATUS='UNKNOWN' )

          OPEN(110,FILE='SPA_cdf.DAT',STATUS='UNKNOWN' )
          OPEN(111,FILE='SPA_cdf_sen_m1.DAT',STATUS='UNKNOWN')
          OPEN(112,FILE='SPA_cdf_sen_m2.DAT',STATUS='UNKNOWN')
          OPEN(113,FILE='SPA_cdf_sen_m3.DAT',STATUS='UNKNOWN')
          OPEN(114,FILE='SPA_cdf_sen_m4.DAT',STATUS='UNKNOWN')
          OPEN(115,FILE='SPA_cdf_sen_m5.DAT',STATUS='UNKNOWN')
          OPEN(116,FILE='SPA_cdf_sen_m6.DAT',STATUS='UNKNOWN')

          OPEN(121,FILE='SPA_cdf_sen_s1.DAT',STATUS='UNKNOWN')
          OPEN(122,FILE='SPA_cdf_sen_s2.DAT',STATUS='UNKNOWN')
          OPEN(123,FILE='SPA_cdf_sen_s3.DAT',STATUS='UNKNOWN')
          OPEN(124,FILE='SPA_cdf_sen_s4.DAT',STATUS='UNKNOWN')
          OPEN(125,FILE='SPA_cdf_sen_s5.DAT',STATUS='UNKNOWN')
          OPEN(126,FILE='SPA_cdf_sen_s6.DAT',STATUS='UNKNOWN')

          OPEN(200,FILE='MCS_cdf.DAT',STATUS='UNKNOWN')

          OPEN(201,FILE='MCS_cdf_sen_m1.DAT',STATUS='UNKNOWN')
          OPEN(202,FILE='MCS_cdf_sen_m2.DAT',STATUS='UNKNOWN')
          OPEN(203,FILE='MCS_cdf_sen_m3.DAT',STATUS='UNKNOWN')
          OPEN(204,FILE='MCS_cdf_sen_m4.DAT',STATUS='UNKNOWN')
          OPEN(205,FILE='MCS_cdf_sen_m5.DAT',STATUS='UNKNOWN')
          OPEN(206,FILE='MCS_cdf_sen_m6.DAT',STATUS='UNKNOWN')

          OPEN(211,FILE='MCS_cdf_sen_s1.DAT',STATUS='UNKNOWN')
          OPEN(212,FILE='MCS_cdf_sen_s2.DAT',STATUS='UNKNOWN')
          OPEN(213,FILE='MCS_cdf_sen_s3.DAT',STATUS='UNKNOWN')
          OPEN(214,FILE='MCS_cdf_sen_s4.DAT',STATUS='UNKNOWN')
          OPEN(215,FILE='MCS_cdf_sen_s5.DAT',STATUS='UNKNOWN')
          OPEN(216,FILE='MCS_cdf_sen_s6.DAT',STATUS='UNKNOWN')

          WRITE(100,*),'cov,       ts_ncum1     ... ts_ncumn'
          WRITE(110,*),'cov,       pf_ncum1     ... pf_ncumn'
          WRITE(200,*),'cov,       pf(crude MCS),        pf(PDD-MCS)'

          do i = 1, N
              WRITE(110+i,*),'cov,       sen_pf_ncum1  ... sen_pf_ncumn'
              WRITE(120+i,*),'cov,       sen_pf_ncum1  ... sen_pf_ncumn'
              WRITE(200+i,*),'cov,       sen_pf(crude MCS),
     1              sen_pf(PDD-MCS)'
              WRITE(210+i,*),'cov,       sen_pf(crude MCS),
     1              sen_pf(PDD-MCS)'
          enddo
       ELSE

          OPEN(90,FILE='Ky.DAT',STATUS='OLD',POSITION='APPEND')

          OPEN(100,FILE='TS.DAT',STATUS='OLD',POSITION='APPEND')

          OPEN(110,FILE='SPA_cdf.DAT',STATUS='OLD',POSITION='APPEND' )
          OPEN(111,FILE='SPA_cdf_sen_m1.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(112,FILE='SPA_cdf_sen_m2.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(113,FILE='SPA_cdf_sen_m3.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(114,FILE='SPA_cdf_sen_m4.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(115,FILE='SPA_cdf_sen_m5.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(116,FILE='SPA_cdf_sen_m6.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')

          OPEN(121,FILE='SPA_cdf_sen_s1.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(122,FILE='SPA_cdf_sen_s2.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(123,FILE='SPA_cdf_sen_s3.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(124,FILE='SPA_cdf_sen_s4.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(125,FILE='SPA_cdf_sen_s5.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(126,FILE='SPA_cdf_sen_s6.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')

          OPEN(200,FILE='MCS_cdf.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')

          OPEN(201,FILE='MCS_cdf_sen_m1.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(202,FILE='MCS_cdf_sen_m2.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(203,FILE='MCS_cdf_sen_m3.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(204,FILE='MCS_cdf_sen_m4.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(205,FILE='MCS_cdf_sen_m5.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(206,FILE='MCS_cdf_sen_m6.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')

          OPEN(211,FILE='MCS_cdf_sen_s1.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(212,FILE='MCS_cdf_sen_s2.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(213,FILE='MCS_cdf_sen_s3.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(214,FILE='MCS_cdf_sen_s4.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(215,FILE='MCS_cdf_sen_s5.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')
          OPEN(216,FILE='MCS_cdf_sen_s6.DAT',STATUS='OLD'
     1     ,POSITION='APPEND')

      ENDIF

c      np = ncum/2
       np = ncum

       WRITE(90,'(f10.6, <np>E20.10)'), cov, Ky(1:np)
       WRITE(90,'(f10.6, <np>F20.10)'), cov, Ky(1:np)

      WRITE(100,'(f10.6, <np>F20.10)'), cov, ts(2:np)
      WRITE(110,'(f10.6, <np>F20.10)'), cov, cdf(2:np)
      WRITE(200,'(f10.6, <2>F20.10)'), cov, PF_MCS, pf_pmcs

      do i = 1, N
         write(110+i,'(f10.6, <np>F20.10)'), cov,cdf_sen(1:np,i,1)
         write(120+i,'(f10.6, <np>F20.10)'), cov,cdf_sen(1:np,i,2)
         WRITE(200+i,'(f10.6, <2>F20.10)'), cov, PF_SEN_MCS(i,1),
     1     pf_sen_pmcs(i,1)
         WRITE(210+i,'(f10.6, <2>F20.10)'), cov, PF_SEN_MCS(i,2),
     1     pf_sen_pmcs(i,2)
         close(110+i)
         close(120+i)
         close(200+i)
         close(210+i)
      enddo
       close(90)
      close(100)
      close(110)
      close(200)

      RETURN

      END
C
C       ------------------------------------------------------------
        SUBROUTINE RESPON(X,Y,IFLAG_Y)
C
C       CALCULATES Y FOR A SAMPLE OF X
C
      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(20),Z(20),XMU(20),SIG(20), NMU(20),NSIG(20)
      REAL*8 Y
      INTEGER NYQ, N
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N
        COMMON /XR01/ N_F, N_GF, N_GC, N_EF,N_EC

      Y = X(1)+ 2*X(2)+ 2*X(3) + X(4) - 5*X(5) - 5*X(6)

        RETURN
        END
C
C---------------------------------------------------------------
        SUBROUTINE SCOR(X,S_MU, S_SIG)
C
C       CALCULATES SCORE FUNCTION FOR A SAMPLE OF X
C
      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(20),Z(20),XMU(20),SIG(20), NMU(20),NSIG(20)
      REAL*8 S_MU(200), S_SIG(200)
        INTEGER NYQ, N
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

C     ONLY FIRST TWO VARIABLES ARE DESIGN VARIABLES, FROM 3-5 VARIABLES, NEED ADD NEW SCORE FUNCTION FOR THIS PROBLEM
      DO I=1,N
      Z(I) = NMU(I) + NSIG(I)*X(I)
      S_MU(I) =  1./NSIG(I) * ((Z(I)-NMU(I))/NSIG(I))
      S_SIG(I) =  1./NSIG(I) * (((Z(I)-NMU(I))/NSIG(I))**2.0  - 1.0)
      ENDDO

        RETURN
        END

C
C---------------------------------------------------------------
        SUBROUTINE SCOR2(S_MU, S_SIG, NGAUSS)
C
C       CALCULATES SCORE FUNCTION FOR AT DIFFERENT GAUSS POINT
C
        USE GQRUL_INT
        USE UMACH_INT

      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(10),Z(10),XMU(20),SIG(20), NMU(20),NSIG(20)
      REAL*8 S_MU(10,10), S_SIG(10,10)
        INTEGER NYQ, N
        REAL*8 QX(50),QW(50),QXFIX(2)
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

        CALL DGQRUL (NGAUSS, 4, 0., 0., 0, QXFIX, QX, QW)
      QX = QX * DSQRT(2.D0)
C     ONLY FIRST TWO VARIABLES ARE DESIGN VARIABLES, FROM 3-5 VARIABLES, NEED ADD NEW SCORE FUNCTION FOR THIS PROBLEM
      DO I=1,N
      DO J=1,NGAUSS
      X(I) = QX(J)
      Z(I) = NMU(I) + NSIG(I)*X(I)
      S_MU(I,J) =  1./NSIG(I) * ((Z(I)-NMU(I))/NSIG(I))
      S_SIG(I,J) =  1./NSIG(I) * (((Z(I)-NMU(I))/NSIG(I))**2.0  - 1.0)
      ENDDO
      ENDDO

        RETURN
        END

C
C---------------------------------------------------------------
        SUBROUTINE PDF_IID(X, Y)
C
C       CALCULATES PDF VALUE FOR A SAMPLE OF X
C
      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(20),Z(20),XMU(20),SIG(20), NMU(20),NSIG(20)
      REAL*8 S_MU, S_SIG
      INTEGER NYQ, N
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

      Y = 1.
      PI = 4.*DATAN(1.0D0)
      DO I = 1,N
        Z(I) = NMU(I) + NSIG(I)*X(I)
        Y = Y * 1./(NSIG(I)*DSQRT(2*PI)) * EXP(-0.5*
     1      ((Z(I)-NMU(I))/NSIG(I))**2)

      ENDDO

        RETURN
        END

C
C       ------------------------------------------------------------
        SUBROUTINE BASIS_NORMAL(X,PSIF,N,NBAS)
C
C       DEFINES HERMITE POLYNOMIALS - NORMALIZED W.R.T. N(0,1) PDF
C       DO NOT INCLUDE THE 0-ORDER, BECAUSE THE 0-ORDER GIVES MEAN
C     ALREADY INCLUDED IN Y0
C
      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(N),PSIF(N,NBAS)
C
        DO I = 1,N
        IF (1 .LE. NBAS) PSIF(I,1) = X(I)
        IF (2 .LE. NBAS) PSIF(I,2) = (X(I)**2 - 1.)/DSQRT(2.D0)
        IF (3 .LE. NBAS) PSIF(I,3) = (X(I)**3 - 3.*X(I))/
     &                               DSQRT(6.D0)
        IF (4 .LE. NBAS) PSIF(I,4) = (X(I)**4 - 6.*X(I)**2 + 3.)/
     &                               DSQRT(24.D0)
        IF (5 .LE. NBAS) PSIF(I,5) = (X(I)**5 - 10.*X(I)**3 + 15.*X(I))/
     &                               DSQRT(120.D0)
        IF (6 .LE. NBAS) PSIF(I,6) = (X(I)**6 - 15.*X(I)**4 + 45.*X(I)**2
     &                              -15.)/DSQRT(720.D0)
        IF (7 .LE. NBAS) PSIF(I,7) = (X(I)**7 - 21.*X(I)**5 + 105.*X(I)**3
     &                              -105.*X(I))/DSQRT(5040.D0)
        IF (8 .LE. NBAS) PSIF(I,8) = (X(I)**8 - 28.*X(I)**6 + 210.*X(I)**4
     &                              -420.*X(I)**2 + 105.)/DSQRT(40320.D0)
        IF (9 .LE. NBAS) PSIF(I,9) = (X(I)**9 - 36.*X(I)**7 + 378.*X(I)**5
     &               -1260.*X(I)**3 + 945.*X(I))/DSQRT(362880.D0)
        ENDDO
C
        RETURN
        END

C
C       ------------------------------------------------------------
        SUBROUTINE BASIS_NORMAL3(X,PSIF,N,NBAS)
C
C       DEFINES ORTHOGONAL POLYNOMIALS - LOG-NORMAL  PDF
C       DO NOT INCLUDE THE 0-ORDER, BECAUSE THE 0-ORDER GIVES MEAN
C     ALREADY INCLUDED IN Y0
C

C   -------------  JUST FOR MEAN = 1050, STD = 250  --------
C   ------ OTHER CASE NEED USING STIELTJES PROCESS RECALCULATE A AND B

      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(N),PSIF(N,NBAS), B(10), A(10)
C

      PSIF = 0.0

      A = 0.0

      B(1) = 1.0D0
      DO I = 2, 10
        B(I) = I-1
      ENDDO

        DO I = 1,N
C       IF (0 .LE. NBAS) PSIF(I,1) = 1.  ----------THIS IS WHY Y0 COME IN
        IF (1 .LE. NBAS) PSIF(I,1) = (X(I) - A(1))/DSQRT(1.0D0*B(2))
        IF (2 .LE. NBAS) PSIF(I,2) = (PSIF(I,1)*(X(I) - A(2))
     1     -DSQRT(1.0D0*B(2)))/DSQRT(1.0D0*B(3))
        IF (3 .LE. NBAS) PSIF(I,3) = ((X(I)-A(3))*PSIF(I,2)-DSQRT(1.0D0*B(3))
     1     *PSIF(I,1))/DSQRT(1.0D0*B(4))

        IF (4 .LE. NBAS) PSIF(I,4) = ((X(I)-A(4))*PSIF(I,3)-DSQRT(1.0D0*B(4))
     1      *PSIF(I,2))/DSQRT(1.0D0*B(5))

        IF (5 .LE. NBAS) PSIF(I,5) = ((X(I)-A(5))*PSIF(I,4)-DSQRT(1.0D0*B(5))
     1      *PSIF(I,3))/DSQRT(1.0D0*B(6))
        IF (6 .LE. NBAS) PSIF(I,6) = ((X(I)-A(6))*PSIF(I,5)-DSQRT(1.0D0*B(6))
     1      *PSIF(I,4))/DSQRT(1.0D0*B(7))
        IF (7 .LE. NBAS) PSIF(I,7) = ((X(I)-A(7))*PSIF(I,6)-DSQRT(1.0D0*B(7))
     1      *PSIF(I,5))/DSQRT(1.0D0*B(8))
        IF (8 .LE. NBAS) PSIF(I,8) = ((X(I)-A(8))*PSIF(I,7)-DSQRT(1.0D0*B(8))
     1      *PSIF(I,6))/DSQRT(1.0D0*B(9))
        IF (9 .LE. NBAS) PSIF(I,9) = ((X(I)-A(9))*PSIF(I,8)-DSQRT(1.0D0*B(9))
     1      *PSIF(I,7))/DSQRT(1.0D0*B(10))
        ENDDO
C
        RETURN
        END

C
C       ------------------------------------------------------------
        SUBROUTINE BASIS_NORMAL2(X,PSIF,NGAUSS,NBAS)
C
C       DEFINES ORTHOGONAL POLYNOMIALS - LOG-NORMAL  PDF
C       DO NOT INCLUDE THE 0-ORDER, BECAUSE THE 0-ORDER GIVES MEAN
C     ALREADY INCLUDED IN Y0
C

C   -------------  JUST FOR MEAN = 1050, STD = 250  --------
C   ------ OTHER CASE NEED USING STIELTJES PROCESS RECALCULATE A AND B

      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 XMU(20),SIG(20), NMU(20),NSIG(20)
      REAL*8 S_MU, S_SIG
      INTEGER NYQ, N
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N
        REAL*8 X(N),PSIF(NBAS,NGAUSS), B(10), A(10)

C

      PSIF = 0.0

        DO I = 1,NGAUSS
        IF (1 .LE. NBAS) PSIF(1,I) = X(I)
        IF (2 .LE. NBAS) PSIF(2,I) = (X(I)**2 - 1.)/DSQRT(2.D0)
        IF (3 .LE. NBAS) PSIF(3,I) = (X(I)**3 - 3.*X(I))/
     &                               DSQRT(6.D0)
        IF (4 .LE. NBAS) PSIF(4,I) = (X(I)**4 - 6.*X(I)**2 + 3.)/
     &                               DSQRT(24.D0)
        IF (5 .LE. NBAS) PSIF(5,I) = (X(I)**5 - 10.*X(I)**3 + 15.*X(I))/
     &                               DSQRT(120.D0)
        IF (6 .LE. NBAS) PSIF(6,I) = (X(I)**6 - 15.*X(I)**4 + 45.*X(I)**2
     &                              -15.)/DSQRT(720.D0)
        IF (7 .LE. NBAS) PSIF(7,I) = (X(I)**7 - 21.*X(I)**5 + 105.*X(I)**3
     &                              -105.*X(I))/DSQRT(5040.D0)
        IF (8 .LE. NBAS) PSIF(8,I) = (X(I)**8 - 28.*X(I)**6 + 210.*X(I)**4
     &                              -420.*X(I)**2 + 105.)/DSQRT(40320.D0)
        IF (9 .LE. NBAS) PSIF(9,I) = (X(I)**9 - 36.*X(I)**7 + 378.*X(I)**5
     &               -1260.*X(I)**3 + 945.*X(I))/DSQRT(362880.D0)
        ENDDO
C
        RETURN
        END

        SUBROUTINE DGQRUL_NORMAL(NGAUSS,QX,QW)
C
C       CALCULATE GAUSS QUADRATURE RULE FOR LOG-NORMAL  PDF (WEIGHT)
C
C
C

C   -------------  JUST FOR MEAN = 1050, STD = 250  --------
C   ------ OTHER CASE NEED USING STIELTJES PROCESS RECALCULATE A AND B

      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 B(10), A(10), QX(10), QW(10)
C

      QX = 0.0
      QW = 0.0

      IF (NGAUSS .EQ. 1) THEN
        QX(1) =  0.0D0
        QW(1) =  1.0D0
      ELSEIF (NGAUSS .EQ. 2) THEN
        QX(1) =  1.0D0
        QW(1) =  0.5D0
        QX(2) =  -1.0D0
        QW(2) =  0.5D0
      ELSEIF (NGAUSS .EQ. 3) THEN
        QX(1) =  -1.7320508076e+00
        QW(1) =  1.6666666667e-01
        QX(2) =  0.0D0
        QW(2) =  6.6666666667e-01
        QX(3) =  1.7320508076e+00
        QW(3) =  1.6666666667e-01
      ELSEIF (NGAUSS .EQ. 4) THEN
        QX(1) =  -2.3344142183e+00
        QW(1) =  4.5875854768e-02
        QX(2) =  -7.4196378430e-01
        QW(2) =  4.5412414523e-01
        QX(3) =  2.3344142183e+00
        QW(3) =  4.5875854768e-02
        QX(4) =  7.4196378430e-01
        QW(4) =  4.5412414523e-01

      ELSEIF (NGAUSS .EQ. 5) THEN
        QX(1) =  -2.8569700139e+00
        QW(1) =  1.1257411328e-02
        QX(2) =  -1.3556261800e+00
        QW(2) =  2.2207592201e-01
        QX(3) =  0.0D0
        QW(3) =  5.3333333333e-01
        QX(4) =  1.3556261800e+00
        QW(4) =  2.2207592201e-01
        QX(5) =  2.8569700139e+00
        QW(5) =  1.1257411328e-02

      ELSEIF (NGAUSS .EQ. 6) THEN
        QX(1) =   -3.3242574336e+00
        QW(1) =    2.5557844021e-03
        QX(2) =   -1.8891758778e+00
        QW(2) =    8.8615746042e-02
        QX(3) =   -6.1670659019e-01
        QW(3) =    4.0882846956e-01
        QX(4) =    6.1670659019e-01
        QW(4) =    4.0882846956e-01
        QX(5) =    1.8891758778e+00
        QW(5) =    8.8615746042e-02
        QX(6) =    3.3242574336e+00
        QW(6) =    2.5557844021e-03

      ELSEIF (NGAUSS .EQ. 7) THEN
        QX(1) =   -3.7504397177e+00
        QX(2) =   -2.3667594107e+00
        QX(3) =   -1.1544053947e+00
        QX(4) =    2.0487226328e-16
        QX(5) =    1.1544053947e+00
        QX(6) =    2.3667594107e+00
        QX(7) =    3.7504397177e+00

        QW(1) =  5.4826885597e-04
        QW(2) =  3.0757123968e-02
        QW(3) =  2.4012317861e-01
        QW(4) =  4.5714285714e-01
        QW(5) =  2.4012317861e-01
        QW(6) =  3.0757123968e-02
        QW(7) =  5.4826885597e-04

      ELSEIF (NGAUSS .EQ. 8) THEN
        QX(1) = -4.1445471861e+00
        QX(2) = -2.8024858613e+00
        QX(3) = -1.6365190424e+00
        QX(4) = -5.3907981135e-01
        QX(5) =  5.3907981135e-01
        QX(6) =  1.6365190424e+00
        QX(7) =  2.8024858613e+00
        QX(8) =  4.1445471861e+00

        QW(1) = 1.1261453838e-04
        QW(2) = 9.6352201208e-03
        QW(3) = 1.1723990766e-01
        QW(4) = 3.7301225768e-01
        QW(5) = 3.7301225768e-01
        QW(6) = 1.1723990766e-01
        QW(7) = 9.6352201208e-03
        QW(8) = 1.1261453838e-04

      ELSE

      PRINT *, 'NGAUSS TOO BIG OR NEGATIVE! '

      STOP

      ENDIF

C
        RETURN
        END

C      MODULE GETAIJK
C      CONTAINS
      SUBROUTINE A_IJK(Y0, ALFA, BITA, GAMA, A, B, NBAS, N, AIJK)
c     1 IFLAG_DIST)
C      SUBROUTINE A_IJK( A, B, NBAS, N, AIJK)
C     SUBROUTINE TO GET COEFICIENTS A_IJK BY PDD COEFs
c     A, B ARE THREE TERMS RECURSION COEFs
C     AIJK X1^I*X2^J*X3^J

      USE UMACH_INT
      USE GAMMA_INT
      REAL*8 Y0, ALFA(10,10),BITA(10,10,10,10),GAMA(10,10,10,10,10,10)
C      REAL*8 Y0
C      REAL, POINTER ::  ALFA(:,:),BITA(:,:,:,:),GAMA(:,:,:,:,:,:)
      REAL*8 A(20,20),B(20,20), AIJK(10,10,10,10,10,10)
      REAL*8 PHI(20,20,20)
      INTEGER NBAS(20), N
C      ALLOCATABLE ALFA, BITA, GAMA

C      COMMON /XR03/ Y0, ALFA, BITA, GAMA

      PHI = 0.
      AIJK = 0.

      DO I = 1, N
      DO J = 1, NBAS(I)
        IF (J .EQ. 1) THEN
            PHI(I,J,1) =  -A(I,J)/DSQRT(1.0D0*B(I,J+1))
            PHI(I,J,2) =  1.0/DSQRT(1.0D0*B(I,J+1))
        ELSEIF (J .EQ. 2) THEN
            PHI(I,J,1) =  (-A(I,J)*PHI(I,J-1,1) -DSQRT(1.0D0*B(I,J)))
     1       /DSQRT(1.0D0*B(I,J+1))
            PHI(I,J,2) =  (PHI(I,J-1,1)-A(I,J)*PHI(I,J-1,2))
     1       /DSQRT(1.0D0*B(I,J+1))
            PHI(I,J,3) =  (PHI(I,J-1,2))
     1       /DSQRT(1.0D0*B(I,J+1))
        ELSE
           PHI(I,J,1) =  (-DSQRT(1.0D0*B(I,J))*PHI(I,J-2,1) -
     1      A(I,J)*PHI(I,J-1,1))/DSQRT(1.0D0*B(I,J+1))
           DO K = 2, J+1
           PHI(I,J,K) =  (-DSQRT(1.0D0*B(I,J))*PHI(I,J-2,K) -
     1      A(I,J)*PHI(I,J-1,K)+ PHI(I,J-1,K-1))/DSQRT(1.0D0*B(I,J+1))
           ENDDO
       ENDIF

      ENDDO
      ENDDO

C     TO DO :  CONSTANT TERM,  1ST ORDER TERM, 2ND ORDER TERM, ....
      AIJK(1,1,1,1,1,1) = Y0;
      DO I = 1, N
      DO J = 1, NBAS(I)
        AIJK(1,1,1,1,1,1) = AIJK(1,1,1,1,1,1) +
     1    ALFA(I,J)*PHI(I,J,1)
        DO K =2, NBAS(I)+1
        AIJK(I,1,1,K,1,1) = AIJK(I,1,1,K,1,1) + ALFA(I,J)*PHI(I,J,K)
        ENDDO

      ENDDO
      ENDDO

      DO I1 = 1, N-1
      DO I2 = I1+1, N
      DO J1 = 1, NBAS(I1)
      DO J2 = 1, NBAS(I2)
        AIJK(1,1,1,1,1,1) = AIJK(1,1,1,1,1,1) +
     1    BITA(I1,I2,J1,J2)*PHI(I1,J1,1)*PHI(I2,J2,1)

        DO I =1, N
        DO K =2, NBAS(I)+1
        IF (I1.EQ.I) THEN
        AIJK(I,1,1,K,1,1) = AIJK(I,1,1,K,1,1) +
     1   BITA(I1,I2,J1,J2)*(PHI(I1,J1,K)*PHI(I2,J2,1))
        ELSEIF (I2.EQ. I) THEN
        AIJK(I,1,1,K,1,1) = AIJK(I,1,1,K,1,1) +
     1   BITA(I1,I2,J1,J2)*(PHI(I1,J1,1)*PHI(I2,J2,K))
        ENDIF

        ENDDO
        ENDDO

        DO K1 =2, NBAS(I1)+1
        DO K2 =2, NBAS(I2)+1
        AIJK(I1,I2,1,K1,K2,1) = AIJK(I1,I2,1,K1,K2,1) +
     1   BITA(I1,I2,J1,J2)*PHI(I1,J1,K1)*PHI(I2,J2,K2)
        ENDDO
        ENDDO

      ENDDO
      ENDDO
      ENDDO
      ENDDO

      DO I1 = 1, N-2
      DO I2 = I1+1, N-1
      DO I3 = I2+1, N
      DO J1 = 1, NBAS(I1)
      DO J2 = 1, NBAS(I2)
      DO J3 = 1, NBAS(I3)
        AIJK(1,1,1,1,1,1) = AIJK(1,1,1,1,1,1) +
     1    GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,1)*PHI(I2,J2,1)*PHI(I3,J3,1)

        DO I =1, N
        DO K =2, NBAS(I)+1
        IF (I1.EQ.I) THEN
        AIJK(I,1,1,K,1,1) = AIJK(I,1,1,K,1,1) +
     1  GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,K)*PHI(I2,J2,1)*PHI(I3,J3,1)
        ELSEIF (I2.EQ. I) THEN
        AIJK(I,1,1,K,1,1) = AIJK(I,1,1,K,1,1) +
     1  GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,1)*PHI(I2,J2,K)*PHI(I3,J3,1)
        ELSEIF (I3.EQ. I) THEN
        AIJK(I,1,1,K,1,1) = AIJK(I,1,1,K,1,1) +
     1  GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,1)*PHI(I2,J2,1)*PHI(I3,J3,K)
        ENDIF

        ENDDO
        ENDDO

        DO II1 = 1,N-1
        DO II2 = II1+1, N
        DO K1 =2, NBAS(II1)+1
        DO K2 =2, NBAS(II2)+1
        IF (I1.EQ.II1.AND. I2.EQ.II2) THEN
        AIJK(II1,II2,1,K1,K2,1) = AIJK(II1,II2,1,K1,K2,1) +
     1  GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,K1)*PHI(I2,J2,K2)*PHI(I3,J3,1)
        ELSEIF (I1.EQ. II1 .AND. I3 .EQ. II2) THEN
        AIJK(II1,II2,1,K1,K2,1) = AIJK(II1,II2,1,K1,K2,1) +
     1  GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,K1)*PHI(I2,J2,1)*PHI(I3,J3,K2)
        ELSEIF (I2.EQ. II1.AND. I3 .EQ.II2) THEN
        AIJK(II1,II2,1,K1,K2,1) = AIJK(II1,II2,1,K1,K2,1) +
     1  GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,1)*PHI(I2,J2,K1)*PHI(I3,J3,K2)
        ENDIF

        ENDDO
        ENDDO
        ENDDO
        ENDDO

      DO K1 =2, NBAS(I1)+1
      DO K2 =2, NBAS(I2)+1
      DO K3 =2, NBAS(I3)+1
      AIJK(I1,I2,I3,K1,K2,K3) = AIJK(I1,I2,I3,K1,K2,K3)+
     1   GAMA(I1,I2,I3,J1,J2,J3)*PHI(I1,J1,K1)*PHI(I2,J2,K2)
     2   *PHI(I3,J3,K3)
      ENDDO
      ENDDO
      ENDDO

      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO

      RETURN
      END

C      END MODULE GETAIJK

      SUBROUTINE EVAL_AIJK(NBAS, AIJK, X, Y)
C     SUBROUTINE TO EVALUATION FUNCTION ACCORDING THE COEFFICIENT A_IJK
C     AIJK X1^I*X2^J*X3^J

      REAL*8 AIJK(10,10,10,10,10,10)
      INTEGER NBAS(20)
C      REAL*8 X(20), Y, Z(20)
      REAL*8 X(20),Z(20),XMU(20),SIG(20), NMU(20),NSIG(20), Y
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

       Z = X

      Y = AIJK(1,1,1,1,1,1)

      DO I = 1,N
      DO K = 2,NBAS(I)+1
      Y = Y + AIJK(I,1,1,K,1,1)*Z(I)**(K-1)
      ENDDO
      ENDDO

      DO I1 = 1,N-1
      DO I2 = I1+1,N
      DO K1 = 2,NBAS(I1)+1
      DO K2 = 2,NBAS(I2)+1
      Y = Y + AIJK(I1,I2,1,K1,K2,1)*Z(I1)**(K1-1)*Z(I2)**(K2-1)
      ENDDO
      ENDDO
      ENDDO
      ENDDO

      DO I1 = 1,N-2
      DO I2 = I1+1,N-1
      DO I3 = I2+1,N
      DO K1 = 2,NBAS(I1)+1
      DO K2 = 2,NBAS(I2)+1
      DO K3 = 2,NBAS(I3)+1
      Y = Y + AIJK(I1,I2,I3,K1,K2,K3)*Z(I1)**(K1-1)*Z(I2)**(K2-1)*
     1   Z(I3)**(K3-1)
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO

      RETURN
      END

      SUBROUTINE RESPON_AIJK(NBAS, AIJK, X, Y)
C     SUBROUTINE TO EVALUATION FUNCTION ACCORDING THE COEFFICIENT A_IJK
C     AIJK X1^I*X2^J*X3^J

      REAL*8 AIJK(10,10,10,10,10,10)
      INTEGER NBAS(20)
C      REAL*8 X(20), Y, Z(20)
      REAL*8 X(20),Z(20),XMU(20),SIG(20), NMU(20),NSIG(20), Y
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

      Z = NMU + NSIG*X

      Y = AIJK(1,1,1,1,1,1)

      DO I = 1,N
      DO K = 2,NBAS(I)+1
      Y = Y + AIJK(I,1,1,K,1,1)*Z(I)**(K-1)
      ENDDO
      ENDDO

      DO I1 = 1,N-1
      DO I2 = I1+1,N
      DO K1 = 2,NBAS(I1)+1
      DO K2 = 2,NBAS(I2)+1
      Y = Y + AIJK(I1,I2,1,K1,K2,1)*Z(I1)**(K1-1)*Z(I2)**(K2-1)
      ENDDO
      ENDDO
      ENDDO
      ENDDO

      DO I1 = 1,N-2
      DO I2 = I1+1,N-1
      DO I3 = I2+1,N
      DO K1 = 2,NBAS(I1)+1
      DO K2 = 2,NBAS(I2)+1
      DO K3 = 2,NBAS(I3)+1
      Y = Y + AIJK(I1,I2,I3,K1,K2,K3)*Z(I1)**(K1-1)*Z(I2)**(K2-1)*
     1   Z(I3)**(K3-1)
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO

      RETURN
      END

C      MODULE GETAIJK
C      CONTAINS
      SUBROUTINE EVAL_PDD(aan,bbn,Y0, ALFA, BITA, GAMA, NBAS, N1, X,Y,
     1 nvar)
c      1 IFLAG_DIST)
C      SUBROUTINE EVAL_PDD(Y0, ALFA, BITA, GAMA, NBAS, N)
C     SUBROUTINE TO GET FUNCTION VALUE BY PDD APPROXIMATION

      USE UMACH_INT
      USE GAMMA_INT
      INTEGER NBAS(20), N1, nvar
      REAL*8 Y0, ALFA(10,10),BITA(10,10,10,10),GAMA(10,10,10,10,10,10)
      REAL*8  H(20,20), PSIF(N1,NBAS(1)),Z(20), XMU(20),SIG(20),
     1   NMU(20),NSIG(20), Y, X(20), aan(10,20), bbn(10,20)

C      COMMON /XR03/ Y0, ALFA, BITA, GAMA
      COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

       Z = X

c      Z(1) =  (X(1)- NMU(1))/NSIG(1)
c      Z(2) =  (X(2)- NMU(2))/NSIG(2)
c      Z =  (X- NMU)/NSIG

C
C       CALCULATE/STORE BASIS FUNCTIONS AT GAUSS POINTS (H)
C     REMARK:  HERE THE DO J SHOULD BE PUT INSIDE AFTER CALL BASIS TO REDUCE COMPUTATION COST

        DO I = 1,N
         DO J = 1,NBAS(I)
         CALL BASIS2(Z,PSIF,N,NBAS(I),aan,bbn)
         H(I,J) = PSIF(I,J)
         ENDDO
        ENDDO

      Y = Y0
      if (nvar .ge. 1) then
      DO I = 1,N
      DO J = 1,NBAS(I)
      Y = Y + ALFA(I,J)*H(I,J)
      ENDDO
      ENDDO
      endif
      if (nvar .ge. 2) then
      DO I1 = 1,N-1
      DO I2 = I1+1,N
      DO J1 = 1,NBAS(I1)
      DO J2 = 1,NBAS(I2)
      Y = Y + BITA(I1,I2,J1,J2)*H(I1,J1)*H(I2,J2)
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      endif

      if (nvar .eq. 3) then
      DO I1 = 1,N-2
      DO I2 = I1+1,N-1
      DO I3 = I2+1,N
      DO J1 = 1,NBAS(I1)
      DO J2 = 1,NBAS(I2)
      DO J3 = 1,NBAS(I3)
      Y = Y + GAMA(I1,I2,I3,J1,J2,J3)*H(I1,J1)*H(I2,J2)*H(I3,J3)
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      ENDDO
      endif

      if (nvar .ne. 1 .and. nvar .ne. 2 .and. nvar .ne. 3) then
        print*, '----Error nvar in eval_pdd!'
        stop
      endif

      RETURN
      END

CCC
CCC

CC

C  -------------- SUBROUTINE TO CALCULATE y univaraite components by dimensions at gauss points AND corresponding weight  at every combination of gauss point
store it to a YU and WU  ------
c
c
c
c
c-------------------------------------------
      SUBROUTINE YWU(N1, NBAS, ALFA)

C     NBASIS(:) : EVERY DIMENSION VALUE SHOULD BE SAME.

      USE UMACH_INT
      USE GAMMA_INT
      INTEGER NBAS(20), N1, NGAUSS, NGAUSS_SPA, NVAR, NBAS_SPA
     1      NYQ, N
      REAL*8 Y0, ALFA(10,10), PI
      REAL*8  H(20,20), PSIF(NBAS(1),NGAUSS_SPA),Z(N1), XMU(20),SIG(20),
     1   NMU(20),NSIG(20), Y, X(20), VBAS(10,10,10), QX(10),QW(10),
     1   QWN(10,10), QXN(10,10), YU(10,10), WU(10,10),QXFIX(2)

C      COMMON /XR03/ Y0, ALFA, BITA, GAMA
      COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N
      COMMON /SPA01/ Y0, YU,WU, NBAS_SPA,  NGAUSS_SPA

      YU = 0.0
      WU = 0.0

      PI = 4.*DATAN(1.0D0)

      DO I = 1, N
        CALL DGQRUL (NGAUSS_SPA,4,0.,0.,0,QXFIX,QX,QW)
c     need double check rxc
        QWN(I,:) = QW/DSQRT(PI)
        QXN(I,:) = QX*DSQRT(2.D0)
        Z = QX(1:NGAUSS_SPA)*DSQRT(2.D0)
        CALL BASIS_NORMAL2 (Z,PSIF,N,NBAS(I))
        VBAS(I,1:NBAS(I),1:NGAUSS_SPA) = PSIF
      ENDDO

      DO I =1,N
      DO J = 1, NBAS(I)
      DO K = 1, NGAUSS_SPA
        YU(I,K) = YU(I,K) + ALFA(I,J)*VBAS(I,J,K)
        WU(I,K) = QWN(I,K)
      ENDDO
      ENDDO
      ENDDO

      RETURN
      END

C  -------------- SUBROUTINE TO CALCULATE MOMENT GENERATING FUNCTION M_Y(t) as a function of t  ------
c
c   calculate all the coefficients of t for univariate PDD of Y only
c
c
c-------------------------------------------
      SUBROUTINE MYT4U(T, YOUT, IDER, IFLAG)

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c                 7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C                 12(KYT''); 20(ALL); 30(Coefficents of Kyt in term of t )

      USE UMACH_INT
      USE GAMMA_INT
c      IMPLICIT NONE
      INTEGER NBAS(20), N1, IFLAG, NVAR, NGAUSS, IDER, NGAUSS_SPA,
     1       NBAS_SPA, NYQ, N
      REAL*8 Y0, YU(10,10), WU(10,10)
      REAL*8  H(20,20),Z(20), XMU(20),SIG(20),
     1   NMU(20),NSIG(20), Y, X(20), VBAS(10,10,10), QWN(10,10),T,
     2  YT, SC(10), QXN(10,10), ETY, YI, ETYI, YOUT(20), ETYI2,
     3   YI2(10), MYT, ETYI0, DKYTN, MYT_SM, D2KYTN, ETYI_SM,
     4   ETYI_SIG,YI_MYT_SM,YI_MYT_SIG, MYT_SIG, YETYI_SM,
     5  YETYI_SIG, YI3(10), D2MYTN_SM,
     6  D2MYTN_SIG, DMYTN_SM, DMYTN_SIG, COE(10,10)

      INTEGER I, I1, I2, I3, IGAUSS,  NBAS_R, NGAUSS_R

      REAL*8 S_MU(10,10), S_SIG(10,10)

C      COMMON /XR03/ Y0, ALFA, BITA, GAMA
      COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

      COMMON /XR04/ NBAS_R, NGAUSS_R, NVAR
      COMMON /SPA01/ Y0, YU,WU, NBAS_SPA, NGAUSS_SPA

      IF (IFLAG .NE. 20 .AND. IFLAG .NE. 1 .AND. IFLAG .NE. 2
     1  .AND. IFLAG .NE. 3 .AND. IFLAG .NE. 4 .AND. IFLAG .NE. 5
     1  .AND. IFLAG .NE. 6 .AND. IFLAG .NE. 7 .AND. IFLAG .NE. 8
     1  .AND. IFLAG .NE. 9 .AND. IFLAG .NE. 10 .AND. IFLAG .NE. 11
     2. .AND. IFLAG .NE. 12) THEN

        PRINT*, "MYT ERROR: WRONG IFLAG VALUE!"
        PRINT*, "IFLAG = ", IFLAG
        STOP
       ENDIF

       NGAUSS = NGAUSS_SPA

      YOUT = 0.0
      COE = 0.0

C     MODIFY HERE
      IF (IFLAG .NE. 20) THEN
        SC = 1.0
      ELSE
        CALL SCOR2(S_MU, S_SIG, NGAUSS_SPA)
      ENDIF

C     ----- CALCUALTE Y AT GAUSS POINTS ------

      Y = Y0
      ETY = EXP(Y0*T)

      IF (NVAR .EQ. 1) THEN

      IF (IFLAG .EQ. 11) THEN
C      ETY = Y0
        DKYTN = 0.0

      DO I = 1,N
        ETYI = 0.0
        ETYI0 = 0.0
      DO IGAUSS = 1, NGAUSS
      ETYI = ETYI + YU(I,IGAUSS)*EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ETYI0 = ETYI0 + EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ENDDO
      DKYTN = DKYTN + ETYI/ETYI0
      ENDDO
      YOUT(11) = DKYTN + Y0

      ENDIF

      IF (IFLAG .EQ. 12) THEN
        D2KYTN = 0.0

      DO I = 1,N
        ETYI = 0.0
        ETYI0 = 0.0
        ETYI2 = 0.0
      DO IGAUSS = 1, NGAUSS
      ETYI = ETYI + YU(I,IGAUSS)*EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ETYI0 = ETYI0 + EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ETYI2 = ETYI2 + YU(I,IGAUSS)**2.0*EXP(YU(I, IGAUSS)*T)*
     1        WU(I,IGAUSS)
      ENDDO
      D2KYTN = D2KYTN + ETYI2/ETYI0 - (ETYI/ETYI0)**2.0
      ENDDO
      YOUT(12) = D2KYTN

      ENDIF

      IF (IFLAG .EQ. 20 .OR. IFLAG .EQ. 30) THEN
      MYT = EXP(T*Y0)
      D2KYTN = 0.0
      DKYTN = Y0
      MYT_SM = EXP(T*Y0)

      DO I = 1,N
        ETYI = 0.0
        ETYI0 = 0.0
        ETYI2 = 0.0
        ETYI_SM = 0.0
        ETYI_SIG = 0.0

      DO IGAUSS = 1, NGAUSS
      ETYI = ETYI + YU(I,IGAUSS)*EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ETYI0 = ETYI0 + EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ETYI2 = ETYI2 + YU(I,IGAUSS)**2.0*EXP(YU(I, IGAUSS)*T)*
     1        WU(I,IGAUSS)
      IF (IDER .EQ. I) THEN
      ETYI_SM = ETYI_SM + EXP(YU(I, IGAUSS)*T)*S_MU(I,IGAUSS)
     1 *WU(I,IGAUSS)
      ETYI_SIG = ETYI_SIG + EXP(YU(I, IGAUSS)*T)*S_SIG(I,IGAUSS)
     1 *WU(I,IGAUSS)
      ELSE
      ELSE
      ETYI_SM = ETYI_SM + EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ETYI_SIG = ETYI_SIG + EXP(YU(I, IGAUSS)*T)*WU(I,IGAUSS)
      ENDIF
      ENDDO
      DKYTN = DKYTN + ETYI/ETYI0
      D2KYTN = D2KYTN + ETYI2/ETYI0 - (ETYI/ETYI0)**2.0
      MYT = MYT*ETYI0

      MYT_SM = MYT_SM * ETYI_SM
      MYT_SIG = MYT_SIG * ETYI_SIG

      ENDDO

      YOUT(1) = MYT
      YOUT(2) = DKYTN * MYT
      YOUT(3) = D2KYTN * MYT  + DKYTN**2.0/MYT
      YOUT(4) = MYT_SM

      YOUT(7) = MYT_SIG
      YOUT(10) = LOG(MYT)
      YOUT(11) = DKYTN
      YOUT(12) = D2KYTN

      DMYTN_SM = Y0 * MYT_SM
      DMYTN_SIG = Y0 * MYT_SIG

      DO I1 = 1,N
      YI_MYT_SM = EXP(T*Y0)
      YI_MYT_SIG = EXP(T*Y0)

      DO I2 = 1,N

        YETYI_SM = 0.0
        YETYI_SIG = 0.0

      IF (I1 .NE. I2) THEN
        YI2 = 1.0
      ELSE
        YI2 = YU(I1,:)
      ENDIF

      DO IGAUSS = 1, NGAUSS

      IF (IDER .EQ. I2) THEN
      YETYI_SM = YETYI_SM + YI2(I2)*EXP(YU(I2, IGAUSS)*T)
     2 *S_MU(I2,IGAUSS) *WU(I2,IGAUSS)
      YETYI_SIG = YETYI_SIG + YI2(I2)*EXP(YU(I2, IGAUSS)*T)
     2 *S_SIG(I2,IGAUSS)*WU(I2,IGAUSS)
      ELSE
      YETYI_SM = YETYI_SM + YI2(I2)*EXP(YU(I2, IGAUSS)*T)*WU(I2,IGAUSS)
      YETYI_SIG= YETYI_SIG + YI2(I2)*EXP(YU(I2, IGAUSS)*T)*WU(I2,IGAUSS)
      ENDIF
      ENDDO

      YI_MYT_SM = YI_MYT_SM * YETYI_SM
      YI_MYT_SIG = YI_MYT_SIG * YETYI_SIG

      ENDDO
      DMYTN_SM = DMYTN_SM + YI_MYT_SM
      DMYTN_SIG = DMYTN_SIG + YI_MYT_SIG

      ENDDO

      YOUT(5) = DMYTN_SM
      YOUT(8) = DMYTN_SIG

      D2MYTN_SM = 2.0*Y0 * DMYTN_SM + Y0**2.0*YOUT(4)
      D2MYTN_SIG = 2.0*Y0 * DMYTN_SIG + Y0**2.0*YOUT(7)

      DO I3 = 1,N
      DO I1 = 1,N
      YI_MYT_SM = EXP(T*Y0)
      YI_MYT_SIG = EXP(T*Y0)

      DO I2 = 1,N

        YETYI_SM = 0.0
        YETYI_SIG = 0.0

      IF (I1 .NE. I2 ) THEN
        YI2 = 1.0
      ELSE
        YI2 = YU(I1,:)
      ENDIF

      IF (I3 .NE. I2 ) THEN
        YI3 = 1.0
      ELSE
        YI3 = YU(I3,:)
      ENDIF

      DO IGAUSS = 1, NGAUSS

      IF (IDER .EQ. I2) THEN
      YETYI_SM = YETYI_SM + YI3(IGAUSS)*YI2(IGAUSS)*
     2        EXP(YU(I2, IGAUSS)*T)*S_MU(I2,IGAUSS) *WU(I2,IGAUSS)
      YETYI_SIG = YETYI_SIG + YI3(IGAUSS)*YI2(IGAUSS)*
     2     EXP(YU(I2, IGAUSS)*T)*S_SIG(I2,IGAUSS)*WU(I2,IGAUSS)
      ELSE
      YETYI_SM = YETYI_SM + YI3(IGAUSS)*YI2(IGAUSS)*
     2        EXP(YU(I2, IGAUSS)*T)*WU(I2,IGAUSS)
      YETYI_SIG= YETYI_SIG + YI3(IGAUSS)*YI2(IGAUSS)*
     2     EXP(YU(I2, IGAUSS)*T)*WU(I2,IGAUSS)
      ENDIF
      ENDDO

      YI_MYT_SM = YI_MYT_SM * YETYI_SM
      YI_MYT_SIG = YI_MYT_SIG * YETYI_SIG

      ENDDO
      D2MYTN_SM = D2MYTN_SM + YI_MYT_SM
      D2MYTN_SIG = D2MYTN_SIG + YI_MYT_SIG

      ENDDO
      ENDDO

      YOUT(6) = D2MYTN_SM
      YOUT(9) = D2MYTN_SIG

      ENDIF

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

      ENDIF

      IF (NVAR .GE. 2 .AND. N .EQ. 2) THEN
      PRINT*, "ERROR, NVAR =2 CALL ANOTHER SUBROUTINE!"
      STOP
      ENDIF

      IF (NVAR .GE. 3) THEN
      PRINT*, "ERROR, NVAR =2 CALL ANOTHER SUBROUTINE!"
      STOP
      ENDIF

      RETURN
      END

C  -------------- SUBROUTINE TO CALCULATE ts  ------
c
c
c
c
c--------------------------------------------
      SUBROUTINE TS(XGUESS1,TS_OUT)

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

      USE ZREAL_INT
      USE WRRRN_INT
! Declare variables

      USE UMACH_INT
      USE GAMMA_INT

C      IMPLICIT NONE

      INTEGER NROOT
      REAL EPS, ERRABS, ERRREL
      PARAMETER (NROOT=1)
!
      INTEGER INFO(NROOT), ITMAX
      REAL F, X(NROOT), XGUESS(NROOT)
      EXTERNAL F

      REAL*8 TS_OUT, XGUESS1
! Set values of initial guess
! XGUESS = ( 4.6 -193.3)
!
      DATA XGUESS/0.0/
!
      XGUESS = 0.0
      EPS = 1.0E-5
      ERRABS = 1.0E-5
      ERRREL = 1.0E-5
      ITMAX = 10000.
! Find the zeros
      CALL ZREAL (F, X, ERRABS=ERRABS, ERRREL=ERRREL, EPS=EPS,
     1  NROOT=NROOT, XGUESS=XGUESS, ITMAX=ITMAX)
!
      CALL WRRRN ("The zeros are", X, 1, NROOT, 1)
!
      TS_OUT = X(1)

      RETURN
      END

C  -------------- SUBROUTINE TO CALCULATE ts  ------
c
c
c
c
c--------------------------------------------
      SUBROUTINE TS2(XGUESS1,TS_OUT)

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

      USE ZBREN_INT
      USE UMACH_INT

      USE UMACH_INT
      USE GAMMA_INT

C      IMPLICIT NONE

! Declare variables
      REAL ERRABS, ERRREL
!
      INTEGER NOUT
! Declare variables

!
      REAL A, B, F
      EXTERNAL F

      REAL*8 TS_OUT, XGUESS1

      A      = 1.0
      B      = -1.0
      ERRABS = 0.0
      ERRREL = 0.000001
      MAXFN  = 10000
      CALL ZBREN (F, A, B, ERRABS=ERRABS, ERRREL=ERRREL, MAXFN=MAXFN)
! Find the zeros

!

!
      TS_OUT = B

      RETURN
      END

C  -------------- SUBROUTINE TO CALCULATE ts  ------
c
c
c
c
c--------------------------------------------
      SUBROUTINE TS3(XGUESS1,TS_OUT)

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

        USE NEQNJ_INT
        USE UMACH_INT

        INTEGER N
        PARAMETER (N=1)

        INTEGER K, NOUT
        REAL FNORM, X(N), XGUESS(N)
        EXTERNAL FCN, LSJAC

        REAL*8 XGUESS1, TS_OUT

        DATA XGUESS/0.0/

        CALL UMACH (2, NOUT)

        CALL NEQNJ (FCN, LSJAC, X, XGUESS=XGUESS, FNORM=FNORM)

        WRITE (NOUT,99999) (X(K),K=1,N), FNORM
99999 FORMAT ('The roots found are', /, ' X = (', 3F5.1,
     1  ')', /, ' with FNORM = ',F5.4, //)

      RETURN
      END

C -------------  FCN -------------

        SUBROUTINE FCN (X, F, N)
        INTEGER N
        REAL X(N), F(N)

        REAL EXP, SIN
        INTRINSIC EXP, SIN
        REAL*8  YOUT(20), X1

!
      X1 = X(1)
      CALL MYT4U (X1, YOUT,  0, 11)

      F(1) = YOUT(11)

        RETURN
        END

        SUBROUTINE LSJAC (N, X, FJAC)
        INTEGER N
        REAL X(N), FJAC(N,N)

        REAL COS, EXP
        INTRINSIC COS, EXP
        REAL*8  YOUT(20), X1

!
        X1 = X(1)

        CALL MYT4U (X1, YOUT,  0, 12)

        FJAC(1,1) = YOUT(12)

        RETURN
        END

C  -------------- SUBROUTINE TO CALCULATE ts  ------
c
c
c
c
c--------------------------------------------
      SUBROUTINE TS4(XGUESS1,TS_OUT)

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

        USE NEQBJ_INT
        USE UMACH_INT

        INTEGER N, ITMAX
        PARAMETER (N=1)

        INTEGER K, NOUT
        REAL X(N), XGUESS(N)
        EXTERNAL FCN2, JAC

        REAL*8 XGUESS1, TS_OUT

        DATA XGUESS/0.0/

        ITMAX = 10000

        CALL UMACH (2, NOUT)

        CALL NEQBJ (FCN2, JAC, X, XGUESS=XGUESS)

        WRITE (NOUT,99999) (X(K),K=1,N)
99999 FORMAT (' The solution to the system is', /, ' X = (', 3F8.3,
     1        ')')
      RETURN
      END

C -------------  FCN -------------

        SUBROUTINE FCN2 (N, X, F)
        INTEGER N
        REAL X(N), F(N)

        REAL EXP, SIN
        INTRINSIC EXP, SIN
        REAL*8  YOUT(20), X1

!
      X1 = X(1)
      CALL MYT4U (X1, YOUT,  0, 11)

      F(1) = YOUT(11)

        RETURN
        END

        SUBROUTINE JAC (N, X, FJAC, LDFJAC)
        INTEGER N, LDFJAC
        REAL X(N), FJAC(LDFJAC,N)

        REAL COS, EXP
        INTRINSIC COS, EXP
        REAL*8  YOUT(20), X1

!
        X1 = X(1)

        CALL MYT4U (X1, YOUT,  0, 12)

        FJAC(1,1) = YOUT(12)

        RETURN
        END

C ---------------- FUNCTION F  -------------
      REAL*8 FUNCTION F (X)
      REAL*8 X, YOUT(20)

!

      CALL MYT4U (X, YOUT,  0, 11)

      F = YOUT(11)

      RETURN
      END

C  -------------- SUBROUTINE TO CALCULATE PF AND ITS SENSITIVITY  ------
c
c
c
c
c--------------------------------------------
      SUBROUTINE PF_SEN_1(TS, Y, PF, PF_SEN)

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

      USE ERFI_INT
      USE ERF_INT
      USE UMACH_INT
      USE GAMMA_INT
      INCLUDE "link_fnl_shared.h"

c      IMPLICIT NONE

      REAL*8 TS, Y, YOUT(20), PF_SEN(10,10), PF, W, V
      REAL*8 MYT, MYT_P, MYT_2P, MYT_SM, MYT_SIG, MYT_P_SM, MYT_2P_SM,
     1    MYT_P_SIG, MYT_2P_SIG, KYT, KYT_P, KYT_2P, KYT_SM,
     2    KYT_SIG, KYT_P_SM, KYT_P_SIG, KYT_2P_SM, KYT_2P_SIG,
     3    W_TS, W_KY, V_TS, V_KY2P, TS_SM, TS_SIG,
     4    W_SM, W_SIG, V_SM, V_SIG,
     5    FY_W, FY_V, PI

      REAL*8   XMU(20),SIG(20),
     1   NMU(20),NSIG(20)
      INTEGER NYQ, N, I

      REAL*8 SIGN_TS, RTMP, RTMP1, RTMP2, RTMP3

      COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

      PI = 4.*DATAN(1.0D0)

      PF = 0.0
      PF_SEN = 0.0

      DO I =1,N

      CALL MYT4U (TS, YOUT,  I, 20)
      MYT = YOUT(1)
      MYT_P  = YOUT(2)
      MYT_2P = YOUT(3)
      MYT_SM = YOUT(4)
      MYT_P_SM  = YOUT(5)
      MYT_2P_SM = YOUT(6)
      MYT_SIG = YOUT(7)
      MYT_P_SIG = YOUT(8)
      MYT_2P_SIG  = YOUT(9)
      KYT = YOUT(10)
      KYT_P = YOUT(11)
      KYT_2P = YOUT(12)

      IF (I .EQ. 1) THEN
         IF (TS .EQ. 0) THEN
            SIGN_TS = 0
         ELSE
            SIGN_TS = TS/ABS(TS)
         ENDIF
         W = SIGN_TS * (2.0*(TS*Y - KYT))**0.5
         V = TS*KYT_2P**0.5
         RTMP = W/DSQRT(2.0D0)
         PF= 0.5D0*(1.0D0 + DERF(RTMP) ) + 1.0/DSQRT(2.0D0*PI)
     1        * EXP(-0.5D0*W**2.0) * (1.0D0/W - 1.0D0/V)
      ENDIF

      KYT_SM = MYT_SM/MYT
      KYT_SIG = MYT_SIG/MYT
      KYT_P_SM = - MYT_SM*MYT_P/(MYT**2.0) + MYT_P_SM/MYT
      KYT_P_SIG = - MYT_SIG*MYT_P/(MYT**2.0) + MYT_P_SIG/MYT
      RTMP1 = 2.0*MYT_P**2.0/(MYT**3.0)
      RTMP2 = 2.0*MYT_P/(MYT**2.0)
      RTMP3 = MYT_2P/(MYT**2.0)

      KYT_2P_SM = MYT_SM*(RTMP1-RTMP3) - RTMP2*MYT_P_SM
     1   + MYT_2P_SM/MYT
      KYT_2P_SIG = MYT_SIG*(RTMP1-RTMP3) - RTMP2*MYT_P_SIG
     1   + MYT_2P_SIG/MYT

      W_TS = Y/W
      W_KY = -1.0/W
      V_TS = KYT_2P**0.5
      V_KY2P = TS/(2.0 * KYT_2P**0.5)
      TS_SM  = -KYT_P_SM/KYT_2P
      TS_SIG  = -KYT_P_SIG/KYT_2P

      W_SM = W_TS*TS_SM + W_KY*KYT_SM
      W_SIG = W_TS*TS_SIG + W_KY*KYT_SIG
      V_SM = V_TS*TS_SM + V_KY2P*KYT_2P_SM
      V_SIG = V_TS*TS_SIG + V_KY2P*KYT_2P_SIG

      FY_W = 1.0/DSQRT(2.0*PI)*EXP(-0.5*W**2.0)
     1        *(W/V - 1.0/(W**2.0))
      FY_V = 1.0/DSQRT(2.0*PI)*EXP(-0.5*W**2.0)/(V**2.0)

      PF_SEN(I,1) = FY_W*W_SM + FY_V*V_SM
      PF_SEN(I,2) = FY_W*W_SIG + FY_V*V_SIG

      ENDDO

      RETURN
      END

C  -------------- SUBROUTINE TO PERFORM DIRECT MCS  ------
c
c
c
c
c--------------------------------------------
      SUBROUTINE DIRECT_MCS(NSAMP, Y_LIM, aan,bbn,Y0, AF1, BT1, GM1,
     1 NBAS_A,NVAR,
     1  PF_MCS, PF_SEN_MCS, pf_pmcs, pf_sen_pmcs)
c     input: NSAMP, Y_LIM, aan,bbn,Y0, AF1, BT1, GM1
C     output:   pf_mcs: crude MCS, pf_sen_mcs: crude MCS, pf_pmcs: pdd MCS, pf_sen_pmcs: pdd MCS
c--- to do
c    1. lognormal random number
c

        USE RNLNL_INT
        USE RNNOF_INT
        USE RNSET_INT
        USE GQRUL_INT
        USE UMACH_INT
        USE GAMMA_INT

        USE DFLIB
C         IMPLICIT NONE
      INCLUDE "link_fnl_shared.h"

      REAL*8 Y, PF_SEN_MCS(20,10), PF_MCS, Y_LIM, pf_sen_pmcs(20,10),
     2 pf_pmcs,Y0

      REAL*8   XMU(20),SIG(20),
     1   NMU(20),NSIG(20)
      INTEGER NYQ, N, NSAMP, KK, KSTEP,ISEED, ISAMP, IND,  IFLAG_Y,
     1  IMU, KKK, I,NBAS_A(20), NVAR
      REAL*8 SMU_A_TMP(200), SSIG_A_TMP(200), X(20), PI, sc(20,10)

      REAL*8 R(1), S, XM, XMN(20), SN(20),aan(10,20), bbn(10,20),
     1 AF1(10,10), BT1(10,10,10,10), GM1(10,10,10,10,10,10),Y_PDD
      COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

        X = 0.
        SN = 0.
        XMN = 0.
        do i =1, N
          SN(i)   = sqrt( log(1+(SIG(i)/XMU(i))**2) )
          XMN(i)  = log(XMU(i)) - 0.5*SN(i)**2
        ENDDO

      PI = 4.*DATAN(1.0D0)

      PF_MCS = 0.0
      PF_SEN_MCS = 0.0
      pf_pmcs = 0.0
      pf_sen_pmcs = 0.0

        KK = 1
      KSTEP = NSAMP/10
        ISEED = 123457

        CALL RNSET(ISEED)

      DO ISAMP =1,NSAMP

         DO I = 1,N
C     This function generates a pseudorandom number from a standard normal distribution.
         CALL RNLNL (XMN(I), SN(I), R)
         X(I) = R(1)
         ENDDO

        IND = 0.
        IFLAG_Y = 1
        CALL RESPON (X,Y, IFLAG_Y)
      call EVAL_PDD(aan,bbn,Y0, AF1, BT1, GM1, NBAS_A, N, X,
     1   Y_PDD, NVAR)

      CALL scor_logn2(X,sc)

        IF (Y .LE. Y_LIM) THEN
        PF_MCS = PF_MCS + 1.0
        DO I =1,N
          PF_SEN_MCS(I, 1) = PF_SEN_MCS(I, 1) + sc(I,1)
          PF_SEN_MCS(I, 2) = PF_SEN_MCS(I, 2) + sc(I,2)
      ENDDO
      ENDIF

        IF (Y_PDD .LE. Y_LIM) THEN
        pf_pmcs = pf_pmcs + 1.0
        DO I =1,N
          pf_sen_pmcs(I, 1) = pf_sen_pmcs(I, 1) + sc(I,1)
          pf_sen_pmcs(I, 2) = pf_sen_pmcs(I, 2) + sc(I,2)
      ENDDO
      ENDIF

C
C       PRINT*, Y0,SYU,SYB,SYT,SYMCS
C

      IF (ISAMP .EQ. KSTEP*KK) THEN
         KKK = KSTEP*KK
        PRINT*, KKK, ' SAMPLES .....'
         KK = KK + 1
      ENDIF

      ENDDO

      PF_MCS = PF_MCS/DFLOAT(NSAMP)
      PF_SEN_MCS = PF_SEN_MCS/DFLOAT(NSAMP)
      pf_pmcs = pf_pmcs/DFLOAT(NSAMP)
      pf_sen_pmcs = pf_sen_pmcs/DFLOAT(NSAMP)

      RETURN
      END

C -------------  PLOT KYT -------------

        SUBROUTINE PLOT_KYT ()
        INTEGER N

        REAL*8  YOUT(20), X1, T,DT, KYT, KYT0,DDT,
     1   KYT_C

!
      T = -10.0
      DT = 0.01
      DDT = 0.01
      KY1 = 508.95830891209531275
      KY2 = 23536.946659474648025
      KY3 = -1.2921168535686265386D06
      KY4 = 1.0113497987868014791D09

      N = 2*ABS(T)/DT

      OPEN(108,FILE='PLOT_KYT.TXT')

      DO I = 0, N
      X1 = T + I*DT

C      CALL MYT4U (X1, YOUT,  1, 20)
      CALL MYT4B(X1,DDT, YOUT, 1, 10)

      KYT = YOUT(10)
C      KYT0 =  X1 + X1**2.0
      KYT_C = KY1 + 0.5*KY2*X1**2 + 1.0/6.0*KY3*X1**3 +
     1      1.0/24.0*KY4*X1**4

      WRITE(108,'(F20.10, F20.10, E20.10)'),
     2      X1, KYT, KYT_C

      ENDDO

      CLOSE(108)
C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

        RETURN
        END

C -------------  PLOT KYT -------------

        SUBROUTINE PLOT_D1KYT ()
        INTEGER N

        REAL*8  YOUT(20), X1, T,DT, KYT, KYT0,DDT,
     1   D1KYT_C

!
      T = -10.0
      DT = 0.01
      DDT = 0.01
      KY1 = 508.95830891209531275
      KY2 = 23536.946659474648025
      KY3 = -1.2921168535686265386D06
      KY4 = 1.0113497987868014791D09

      N = 2*ABS(T)/DT

      OPEN(108,FILE='PLOT_D1KYT.TXT')

      DO I = 0, N
      X1 = T + I*DT
C     CALL MYT4U (X1, YOUT,  1, 11)
      CALL MYT4B(X1,DDT, YOUT, 1, 10)

      D1KYT = YOUT(11)
C      D1KYT0 =  1 + X1*2.0
      D1KYT_C =  KY2*X1 + 0.5*KY3*X1**2 +
     1      1.0/6.0*KY4*X1**3

      WRITE(108,'(F20.10, F20.10, E20.10)'),
     2      X1, D1KYT, D1KYT_C

      ENDDO

      CLOSE(108)
C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL)

        RETURN
        END

C  -------------- SUBROUTINE TO CALCULATE MOMENT GENERATING FUNCTION M_Y(t) as a function of t  ------
c
c   calculate all the coefficients of t for univariate PDD of Y only
c
c
c--------------------------------------------
      SUBROUTINE MYT4B(T,DT, YOUT, IDER, IFLAG)

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL); 30(Coefficents of Kyt in term of t )

        USE GQRUL_INT

      USE UMACH_INT
      USE GAMMA_INT
c      IMPLICIT NONE
      INTEGER NBAS(20), N1, IFLAG, NVAR, NGAUSS, IDER, NGAUSS_SPA,
     1       NBAS_SPA, NYQ, N
      REAL*8 Y0, YU(10,10), WU(10,10), Y_PDD, DT, TDT
      REAL*8  H(20,20),Z(20), XMU(20),SIG(20),
     1   NMU(20),NSIG(20), Y, X(20), VBAS(10,10,10),T,
     2  YT, SC(10),  ETY, YI, ETYI, YOUT(20), ETYI2,
     3   YI2(10), MYT, ETYI0, DKYTN, MYT_SM, D2KYTN, ETYI_SM,
     4   ETYI_SIG,YI_MYT_SM,YI_MYT_SIG, MYT_SIG, YETYI_SM,
     5  YETYI_SIG, YI3(10), D2MYTN_SM,
     6  D2MYTN_SIG, DMYTN_SM, DMYTN_SIG, COE(10,10),
     7  AF1(10,10), BT1(10,10,10,10),QXFIX(2),QXFIX_SM(2),
     8  QX(50),QW(50), QX_SM(50),QW_SM(50),
     9  QXN(20,50),QWN(20,50),XXN(20,50),
     1  QXN_SM(20,50),QWN_SM(20,50),XXN_SM(20,50),
     2  GM1(10,10,10,10,10,10), TMP, MYTDT, KYTDT, KYT, D1KYT

      INTEGER I, I1, I2, I3, IGAUSS,  NBAS_R, NGAUSS_R
      DOUBLE PRECISION PI

      REAL*8 S_MU(10,10), S_SIG(10,10)

C      COMMON /XR03/ Y0, ALFA, BITA, GAMA
      COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

      COMMON /XR04/ NBAS_R, NGAUSS_R, NVAR
      COMMON /SPA01/ Y0, YU,WU, NBAS_SPA, NGAUSS_SPA

      COMMON /SPA02/ AF1, BT1, GM1

      IF (IFLAG .NE. 20 .AND. IFLAG .NE. 1 .AND. IFLAG .NE. 2
     1  .AND. IFLAG .NE. 3 .AND. IFLAG .NE. 4 .AND. IFLAG .NE. 5
     1  .AND. IFLAG .NE. 6 .AND. IFLAG .NE. 7 .AND. IFLAG .NE. 8
     1  .AND. IFLAG .NE. 9 .AND. IFLAG .NE. 10 .AND. IFLAG .NE. 11
     2. .AND. IFLAG .NE. 12) THEN

        PRINT*, "MYT ERROR: WRONG IFLAG VALUE!"
        PRINT*, "IFLAG = ", IFLAG
        STOP
       ENDIF

       NGAUSS = NGAUSS_SPA

      YOUT = 0.0
      COE = 0.0
      NBAS = NBAS_R
      GAMA = 0.
      TDT = T + DT
      PI = 4.*DATAN(1.0D0)

C     MODIFY HERE
      IF (IFLAG .NE. 20) THEN
        SC = 1.0
      ELSE
        CALL SCOR2(S_MU, S_SIG, NGAUSS_SPA)
      ENDIF

C     get GQ rule
        CALL DGQRUL (NGAUSS_SPA,4,0.,0.,0,QXFIX,QX,QW)
        CALL DGQRUL (NGAUSS_SPA,4,0.,0.,0,QXFIX_SM,QX_SM,QW_SM)
        DO J = 1, N
      DO I = 1,NGAUSS_SPA
        QXN(J,I) = QX(I)*DSQRT(2.D0)
        QWN(J,I) = QW(I)/DSQRT(PI)
        XXN(J,I) = QX(I)
        ENDDO
        DO I =1, NGAUSS_SPA
        QXN_SM(J,I) = QX_SM(I)*DSQRT(2.D0)
        QWN_SM(J,I) = QW_SM(I)/DSQRT(PI)
      XXN_SM(J,I)  = QX_SM(I)
        ENDDO
        ENDDO

C     calculate Y_pdd
      X = 0.
      MYT = 0.
      MYTDT = 0.
      DO I1 = 1, N-1
      DO I2 = I1+1, N

      DO J1 = 1,NGAUSS_SPA
      DO J2 = 1,NGAUSS_SPA

      X(I1) = QXN(I1,J1)
      X(I2) = QXN(I2,J2)

c      CALL EVAL_PDD(Y0, AF1, BT1, GM1, NBAS, N, X, Y_PDD)
C     DOUBLE CHECK QX, QW IS RIGHT FOR THIS KIND OF GQ  --- RIGHT!
      TMP = QWN(I1,J1)* QWN(I2,J2)
      MYT = MYT + EXP(T*Y_PDD)*TMP
      MYTDT = MYTDT + EXP(TDT*Y_PDD)*TMP

      ENDDO
      ENDDO

      ENDDO
      ENDDO

      KYT = LOG(MYT)
      KYTDT = LOG(MYTDT)
      D1KYT = (KYTDT - KYT)/DT

      YOUT = 0.0

      YOUT(10) = KYT
      YOUT(11) = D1KYT

C     IFLAG = 1(MYT); 2(MYT'); 3(MYT''); 4(P(MYT)/P(MU)); 5(P(MYT')/P(MU)); 6(P(MYT'')/P(MU));
c             7(P(MYT)/P(SIG)); 8(P(MYT')/P(SIG)); 9(P(MYT'')/P(SIG)); 10(KYT); 11(KYT');
C             12(KYT''); 20(ALL); 30(Coefficents of Kyt in term of t )

      RETURN
      END

C-------------------------------------------------------

      subroutine DGQRUL_LOGN2(mu, sig, i_flag, nbasis,
     1  a, b, ngauss, qx, qw )

c-------------------------------------------------------
C     produce monic poly nomial recursion coef and gauss quadrature rule
c             for lognormal distribution
c
C     input:    mu, sig, i_flag, nbasis
C     output:    a, b, ngauss, qx, qw
```

```
C     output:    a, b, ngauss, qx, qw
c
C     i_flag = 0; mu = MEAN OF LOGNORMAL,  sig = STANDARD DEVIATION OF LOGNORMAL
c                            --- need do a transformation
c
C     i_flag = 1; mu = shape paramerter ,  sig = shape paramerter
C
C------------------------------------------------------------

      IMPLICIT NONE
      REAL*8 b(20), a(20), qx(50,50), qw(50,50), mu, sig, mut, sigt,
     1 lambda, zeta, qx1(50), qw1(50)
      INTEGER ngauss, i_flag, nbasis, i

      a = 0.
      b = 0.
      qx = 0.
      qw = 0.

      if (i_flag .EQ. 1) then
        mut = mu
        sigt = sig
      elseif (i_flag .eq. 0) then
        sigt = dsqrt(log(1+(sig/mu)**2.0))
        mut = log(mu) - 0.5*sigt**2.0
      else
        print*,'wrong i_flag in DGQRUL_LOGN2, i_flag should be 0 or 1!'
        stop
      endif

      zeta = exp(sigt**2.0)
      lambda = exp(mut)

      a(1) = lambda*zeta**((2.0*0-1)/2.0)*( zeta**0*(zeta+1.0)-1.0)
      b(1) = 1.0

      do i = 1, ngauss-1
        a(i+1) = lambda*zeta**((2.0*i-1)/2.0)*( zeta**i*(zeta+1.0)-1.0)
        b(i+1) = lambda**2.0*zeta**(3.0*i-2.0)*(zeta**i-1.0);
      enddo

      if (ngauss .gt. 50) then
         print*, 'Error! ngauss in DGQRUL_LOGN2 great than 50!'
         stop
      endif
      if (ngauss .lt. nbasis) then
         print*, 'Error! ngauss in DGQRUL_LOGN2 less than nbasis!'
         stop
       endif

      do i = 1, ngauss
      call gauss2(i, a, b, qx1, qw1)
      qx(i,:) = qx1(:)
      qw(i,:) = qw1(:)
      enddo

      END

C-----------------------------------------------------------

      subroutine gauss2(ngauss,a, b, qx, qw)
c-----------------------------------------------------------
C     produce gauss quadrature rule from recursion coefficient a, b
c               by solving a eigenvalue problem
C     input:    ngauss,a, b,
C     output:   qx, qw
c
C
C------------------------------------------------------------

      USE EVCSB_INT
      USE EPISB_INT
      USE UMACH_INT
      USE WRRRN_INT

      INCLUDE "link_fnl_shared.h"

      IMPLICIT NONE
      REAL*8 b(20), a(20), qx(50),qw(50)
      INTEGER ngauss, i, j

      INTEGER LDA, LDEVEC, N, NCODA, IWK(ngauss)
      PARAMETER (NCODA=1, LDA=NCODA+1)
      INTEGER NOUT
      REAL*8 AA(LDA,ngauss), EVAL(ngauss), EVEC(ngauss,ngauss), PI,
     1 AA2(1,1), ACOPY( ngauss*NCODA),WK(ngauss)

      N = ngauss
      LDEVEC = ngauss
c Define values of A:
c AA = ( 5 -4 1 )
c ( -4 6 -4 1 )
c ( 1 -4 6 -4 1 )
c ( 1 -4 6 -4 1 )
c ( 1 -4 6 -4 )
c ( 1 -4 5 )
c Represented in band symmetric
c form this is:
c AA = ( 0 0 1 1 1 1 )
c     ( 0 -4 -4 -4 -4 -4 )
c     ( 5 6 6 6 6 5 )
c
c      DATA AA/0.0, 0.0, 5.0, 0.0, -4.0, 6.0, 1.0, -4.0, 6.0, 1.0, -4.0,
c     1 6.0, 1.0, -4.0, 6.0, 1.0, -4.0, 5.0/
      qx = 0.
      qw = 0.
      if (ngauss .gt. 1) then
      do j = 2,ngauss
        AA(1,j) = dsqrt(b(j))
      enddo
      AA(1,1) = 0.

      do j = 1,ngauss
        AA(2,j) = a(j)
      enddo
C Find eigenvalues and vectors
c      CALL DE4CSB (N, AA, LDA, NCODA, EVAL, EVEC, LDEVEC, ACOPY, WK,IWK)
      CALL EVCSB (AA, NCODA, EVAL, EVEC)
C Compute performance index
      PI = EPISB(ngauss,AA,NCODA,EVAL,EVEC)

      else
        AA2(1,1) = A(1)
C Find eigenvalues and vectors
c      CALL DE4CSB (N, AA2, LDA, NCODA, EVAL, EVEC, LDEVEC,
c     1    ACOPY, WK,IWK)
      CALL EVCSB (AA2, 0, EVAL, EVEC)
C Compute performance index
      PI = EPISB(ngauss,AA2,0,EVAL,EVEC)

      endif

C Print results
      CALL UMACH (2, NOUT)
      CALL WRRRN ('EVAL', EVAL, 1, ngauss, 1)
      CALL WRRRN ('EVEC', EVEC)
      WRITE (NOUT,'(/,A,F6.3)') ' Performance index = ', PI

      do i = 1,ngauss
        qx(i) = EVAL(ngauss-i+1)
        qw(i) = b(1)*EVEC(1, ngauss-i+1)**2.0
      enddo

      end

C
C       -------------------------------------------------------------
        SUBROUTINE BASIS2(X,PSIF,N,NBAS,A,B)
C

C       DEFINES ORTHOGONAL POLYNOMIALS up to order 9 by input of recursion coeff, a, b
C       DO NOT INCLUDE THE 0-ORDER, BECAUSE THE 0-ORDER GIVES MEAN
C     ALREADY INCLUDED IN Y0
C-----------------------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(N),PSIF(N,NBAS), A(10,20), B(10,20)
C

        DO I = 1,N
C       IF (0 .LE. NBAS) PSIF(I,1) = 1.  ----------THIS IS WHY Y0 COME IN
        IF (1 .LE. NBAS) PSIF(I,1) = (X(I) - A(I,1))/DSQRT(1.0D0*B(I,2))
        IF (2 .LE. NBAS) PSIF(I,2) = (PSIF(I,1)*(X(I) - A(I,2))
     1     -DSQRT(1.0D0*B(I,2)))/DSQRT(1.0D0*B(I,3))
        IF (3 .LE. NBAS) PSIF(I,3) = ((X(I)-A(I,3))*PSIF(I,2)
     1     -DSQRT(1.0D0*B(I,3))*PSIF(I,1))/DSQRT(1.0D0*B(I,4))

        IF (4 .LE. NBAS) PSIF(I,4) = ((X(I)-A(I,4))*PSIF(I,3)
     1      -DSQRT(1.0D0*B(I,4)) *PSIF(I,2))/DSQRT(1.0D0*B(I,5))

        IF (5 .LE. NBAS) PSIF(I,5) = ((X(I)-A(I,5))*PSIF(I,4)
     1      -DSQRT(1.0D0*B(I,5)) *PSIF(I,3))/DSQRT(1.0D0*B(I,6))
        IF (6 .LE. NBAS) PSIF(I,6) = ((X(I)-A(I,6))*PSIF(I,5)
     1      -DSQRT(1.0D0*B(I,6)) *PSIF(I,4))/DSQRT(1.0D0*B(I,7))
        IF (7 .LE. NBAS) PSIF(I,7) = ((X(I)-A(I,7))*PSIF(I,6)
     1      -DSQRT(1.0D0*B(I,7)) *PSIF(I,5))/DSQRT(1.0D0*B(I,8))
        IF (8 .LE. NBAS) PSIF(I,8) = ((X(I)-A(I,8))*PSIF(I,7)
     1      -DSQRT(1.0D0*B(I,8)) *PSIF(I,6))/DSQRT(1.0D0*B(I,9))
        IF (9 .LE. NBAS) PSIF(I,9) = ((X(I)-A(I,9))*PSIF(I,8)
     1      -DSQRT(1.0D0*B(I,9)) *PSIF(I,7))/DSQRT(1.0D0*B(I,10))
        ENDDO
C
        RETURN
        END

C
C       -------------------------------------------------------------
        SUBROUTINE scor_logn2(X,sc)
C
C       CALCULATES SCORE FUNCTION of lognormal distribution FOR A SAMPLE OF X
C
      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 X(20),Z(20),XMU(20),SIG(20), NMU(20),NSIG(20)
      REAL*8 S_MU(200), S_SIG(200), sc(20,10),sigt(20), mut(20),
     1  dst_m(20), dst_s(20), dmt_m(20), dmt_s(20)
        INTEGER NYQ, N
        COMMON /SR02/ XMU,SIG, NMU, NSIG, NYQ, N

C
      sc = 0.
      sigt = 0.
      mut = 0.
      Z(1:N) = X(1:N)

      DO I=1,N
      sigt(I) = dsqrt(log(1+(NSIG(I)/NMU(I))**2.0))
      mut(I) = log(NMU(I)) - 0.5*sigt(I)**2.0
      dst_m(I) = -1.0/(sigt(I)*(1+(NSIG(I)/NMU(I))**2.0)) *
     1  NSIG(I)**2.0/NMU(I)**3.0
      dst_s(I) = 1.0/(sigt(I)*(1+(NSIG(I)/NMU(I))**2.0)) *
     1  NSIG(I)/NMU(I)**2.0
      dmt_m(I) = 1.0/NMU(I) - sigt(I)*dst_m(I)
      dmt_s(I) =  - sigt(I)*dst_s(I)

      sc(I,1) = -1.0/sigt(I)* dst_m(I) + 1.0/sigt(I)**3.0 *(log(Z(I))
     1 - mut(I)) * (sigt(I) *dmt_m(I) + (log(Z(I))-mut(I))* dst_m(I))
      sc(I,2) = -1.0/sigt(I)* dst_s(I) + 1.0/sigt(I)**3.0 *(log(Z(I))
     1 - mut(I)) * (sigt(I) *dmt_s(I) + (log(Z(I))-mut(I))* dst_s(I))

      ENDDO

        RETURN
        END

c------------------------------------------
      SUBROUTINE ts_spa0(ncum, cum, y, ts)
c   ******  aproximate CGF by trucated poly coefficiented by cumulants ******
c     input: cum,   several cumulants (even number of cumulants)
c     output    :    ts : the saddle point, ts(i) saved the SP by 2i cumulants
c   ****************************************

      USE ZPLRC_INT
      USE FAC_INT
      USE UMACH_INT
      USE WRCRN_INT
      INTEGER NDEG, ncum, i, j, nreal
      real*8  cum(20), ts(20), COEFF(20), c(20), y, d, min_d
      COMPLEX*16 ZERO(20), z(20)

      COEFF = 0.
      c = 0.
      ZERO = 0.
      z = 0.

      ts = 0.

      ts(1) = (y-cum(1))/cum(2)

      do i = 1,ncum
        COEFF(i) = cum(i)/DFAC(i-1)
c      COEFF(i) = FAC(i-1)
      enddo

      do i = 4, ncum, 2
        nreal = 0
        NDEG = i-1
        c(1:20) = COEFF(1:20)
        CALL ZPLRC (c, z, NDEG)
        CALL WRCRN ('The zeros found are', z, 1, NDEG, 1)
        min_d = 10.0**10
        do j = 1, NDEG
            if (AIMAG(z(j)) .eq. 0 .and. REAL(z(j)) .le. 1.D38) then
                nreal = nreal + 1
                d = ABS(REAL(z(j)) - ts(i/2 - 1))
                if (d .lt. min_d ) then
                    min_d = d
                    ts(i/2) = REAL(z(j))
                endif
            endif
        enddo

        if (nreal .eq. 0) then
            print*, 'Error : No real solution found!'
            stop
        endif

      enddo

      RETURN
      END

c------------------------------------------
      SUBROUTINE ts_spa(ncum, cum, y, ts)
c   ******  aproximate CGF by trucated poly coefficiented by cumulants ******
c     input: cum,   several cumulants
c     output    :    ts : the saddle point, ts(i) saved the SP by i cumulants
c                    ts(1)      :       not used, because 1 cumulant will not work for anything
c   ****************************************

      USE ZPLRC_INT
      USE FAC_INT
      USE UMACH_INT
      USE WRCRN_INT
      INTEGER NDEG, ncum, i, j, nreal
      real*8  cum(20), ts(20), COEFF(20), c(20), y, d, min_d,
     1  ts_small
      COMPLEX*16 ZERO(20), z(20)

      COEFF = 0.
      c = 0.
      ZERO = 0.
      z = 0.

      ts = 0.

      ts(2) = (y-cum(1))/cum(2)

      do i = 1,ncum
        COEFF(i) = cum(i)/DFAC(i-1)
      enddo

      do i = 3, ncum, 1
        nreal = 0
        NDEG = i-1
        c(1:20) = COEFF(1:20)
        CALL ZPLRC (c, z, NDEG)
        CALL WRCRN ('The zeros found are', z, 1, NDEG, 1)
        min_d = 10.0**10
        ts_small = 10.0**15
        do j = 1, NDEG
            if (AIMAG(z(j)) .eq. 0 .and. REAL(z(j)) .le. 1.D38) then
                nreal = nreal + 1
                d = ABS(REAL(z(j)) - ts(i - 1))
                if (d .lt. min_d ) then
                    min_d = d
                    ts(i) = REAL(z(j))
                endif
C                 if (REAL(z(j)) .lt. ts_small) then
C                    ts_small = REAL(z(j))
C                    ts(i) = REAL(z(j))
C                 endif
            endif
        enddo

        if (nreal .eq. 0) then
            print*, 'Error : No real solution found!'
            stop
        endif

      enddo

      RETURN
      END

c------------------------------------------
      SUBROUTINE cdf_spa0(ncum, cum, y, ts, cdf)
c   ******  comput cdf from saddle points (even cumulants)******
c     input: cum, ncum, y, ts   several cumulants, y : is the limitation of the performance function
c     output    :    cdf  : cdf at y, cdf(i) saved the cdf by 2i cumulants
c   ****************************************

      USE FAC_INT
      USE ERFI_INT
      USE ERF_INT

      INTEGER ncum, i, j
      real*8  cum(20), ts(20),y, cdf(20), w(20), v(20),
     1  Ky(20), d2Ky(20), PI

      PI = 4.*DATAN(1.0D0)
      cdf = 0.
      w = 0.
      v = 0.
      Ky = 0.
      d2Ky = 0.

      do i = 2, ncum, 2
        if (i .eq. 2) then
           Ky(i/2) = cum(i)/DFAC(i)*ts(i/2)**(i)
     1        + cum(i-1)/DFAC(i-1)*ts(i/2)**(i-1)
           d2Ky(i/2) = cum(i)/DFAC(i-2)*ts(i/2)**(i-2)
        else
c       error here ----------- need modify if use this function
           Ky(i/2) = Ky(i/2-1) + cum(i)/DFAC(i)*ts(i/2)**(i)
     1        + cum(i-1)/DFAC(i-1)*ts(i/2)**(i-1)
            d2Ky(i/2) = d2Ky(i/2-1) + cum(i)/DFAC(i-2)*ts(i/2)*(i-2)
     1        + cum(i-1)/DFAC(i-3)*ts(i/2)**(i-3)
        endif
        if (ts(i/2) .ne. 0) then
            w(i/2) = ts(i/2)/ABS(ts(i/2)) * (2*(ts(i/2)*y - Ky(i/2)))
     1          **0.5
        endif
        v(i/2) = ts(i/2)* d2Ky(i/2)**0.5
        if (w(i/2) .eq. 0 .or. v(i/2) .eq. 0) then
            cdf(i/2) = 0.5*(1.0 + ERF(w(i/2)/sqrt(2.0)))
        else
            cdf(i/2) = 0.5*(1.0 + ERF(w(i/2)/sqrt(2.0))) +
     1       1.0/sqrt(2.0*PI)*exp(-w(i/2)**2.0/2.0)
     1        *(1.0/w(i/2)-1.0/v(i/2))
        endif
      enddo

      RETURN
      END

c------------------------------------------
      SUBROUTINE cdf_spa(ncum, cum, y, ts, cdf, Ky)
c   ******  comput cdf from saddle points ******
c     input: cum, ncum, y, ts   several cumulants, y : is the limitation of the performance function
c     output    :    cdf  : cdf at y, cdf(i) saved the cdf by 2i cumulants
c                    Ky,   output for testing value of Ky
c   ****************************************

      USE FAC_INT
      USE ERFI_INT
      USE ERF_INT

      INTEGER ncum, i, j
      real*8  cum(20), ts(20),y, cdf(20), w(20), v(20),
     1  Ky(20), d2Ky(20), PI

      PI = 4.*DATAN(1.0D0)
      cdf = 0.
      w = 0.
      v = 0.
      Ky = 0.
      d2Ky = 0.

      do i = 1, ncum, 1
        if (i .eq. 1) then
           Ky(i) = cum(i)/DFAC(i)*ts(i)**(i)
           d2Ky(i) = 0.
        else
           do j = 1, i
           Ky(i) = Ky(i) + cum(j)/DFAC(j)*ts(i)**(j)
           enddo
           do j = 2, i
           d2Ky(i) = d2Ky(i) + cum(j)/DFAC(j-2)*ts(i)**(j-2)
           enddo
        endif
        if (ts(i) .ne. 0) then
            w(i) = ts(i)/ABS(ts(i)) * (2*(ts(i)*y - Ky(i)))
     1          **0.5
        endif
        v(i) = ts(i)* d2Ky(i)**0.5
        if (w(i) .eq. 0 .or. v(i) .eq. 0) then
            cdf(i) = 0.5*(1.0 + ERF(w(i)/sqrt(2.0)))
        else
            cdf(i) = 0.5*(1.0 + ERF(w(i)/sqrt(2.0))) +
     1       1.0/sqrt(2.0*PI)*exp(-w(i)**2.0/2.0)
     1        *(1.0/w(i)-1.0/v(i))
        endif
      enddo

      RETURN
      END

c------------------------------------------
      SUBROUTINE sen_cdf_spa0(N,ncum,ntheta, cum, cum_sen, y, ts, cs)
c   ******  comput cdf from saddle points ****** (using even cumulants)
c     input: cum, ntheta, ncum, cum_sen, y, ts   several cumulants, y : is the limitation of the performance function
c     output    :    cs  : sensitivity of cdf at y, cs(i) saved the sensitivity of cdf by 2i cumulants
c               cs(cum, N, ntheta)
c     c     ntheta is the # of distribution parameters of each variable
c           c       cum_sen(ncum,N,ntheta)
c   ****************************************

      USE FAC_INT
      USE ERFI_INT
      USE ERF_INT

      INTEGER ncum, i, j
      real*8  cum(20), ts(20),y, cs(20,20,10), w(20), v(20),
     1  Ky(20), d2Ky(20), PI , cum_sen(20,20,10), d3Ky(20),
     2  Ky_dc(20,20), d1Ky_dc(20,20), d2Ky_dc(20,20)
c       Ky_dc(ncum, derivative to ith cum)
      real*8  Pf_w(20), Pf_v(20), w_c(20,20),v_c(20,20),
     1      w_t(20), v_t(20), w_Ky(20), v_d2Ky(20), ts_c(20,20),
     2      fai(20), tmp, cdf(20)

      PI = 4.*DATAN(1.0D0)
      cs = 0.
      w = 0.
      v = 0.
      Ky = 0.
      d2Ky = 0.
      d3Ky = 0.
      Ky_dc = 0.
      d1Ky_dc = 0.
      d2Ky_dc = 0.
      Pf_w = 0.
      Pf_v = 0.
      w_c = 0.
      v_c = 0.
      w_t = 0.
      v_t=0.
      w_Ky = 0.
      v_d2Ky =0.
      ts_c = 0.
      fai(20) = 0.

      do i = 2, ncum, 2
        do j = 1, i
           Ky_dc(i/2,j) = 1.0/DFAC(j)*ts(i/2)**j
           d1Ky_dc(i/2,j) = 1.0/DFAC(j-1)*ts(i/2)**(j-1)
        enddo
        do j = 2, i
           d2Ky_dc(i/2,j) = 1.0/DFAC(j-2)*ts(i/2)**(j-2)
        enddo

        if (i .eq. 2) then
           Ky(i/2) = cum(i)/DFAC(i)*ts(i/2)**(i)
     1        + cum(i-1)/DFAC(i-1)*ts(i/2)**(i-1)
           d2Ky(i/2) = cum(i)/DFAC(i-2)*ts(i/2)**(i-2)

        else
           Ky(i/2) = Ky(i/2-1) + cum(i)/DFAC(i)*ts(i/2)**(i)
     1        + cum(i-1)/DFAC(i-1)*ts(i/2)**(i-1)
            d2Ky(i/2) = d2Ky(i/2-1) + cum(i)/DFAC(i-2)*ts(i/2)*(i-2)
     1        + cum(i-1)/DFAC(i-3)*ts(i/2)**(i-3)

            d3Ky(i/2) = d3Ky(i/2-1) + cum(i)/DFAC(i-3)*ts(i/2)*(i-3)
     1        + cum(i-1)/DFAC(i-4)*ts(i/2)**(i-4)

        endif
        if (ts(i/2) .ne. 0) then
            w(i/2) = ts(i/2)/ABS(ts(i/2)) * (2*(ts(i/2)*y - Ky(i/2)))
     1          **0.5
        endif
        v(i/2) = ts(i/2)* d2Ky(i/2)**0.5
        fai(i/2) = 1.0/sqrt(2.0*PI)*exp(-w(i/2)**2.0/2.0)
        if (w(i/2) .eq. 0 .or. v(i/2) .eq. 0) then
            cdf(i/2) = 0.5*(1.0 + ERF(w(i/2)/sqrt(2.0)))
        else
            cdf(i/2) = 0.5*(1.0 + ERF(w(i/2)/sqrt(2.0))) +
     1       1.0/sqrt(2.0*PI)*exp(-w(i/2)**2.0/2.0)
     1        *(1.0/w(i/2)-1.0/v(i/2))
        endif
      enddo

c     **********   calculate sensitivity of cdf   ************

      do i = 2, ncum, 2
         if (w(i/2) .eq. 0) then
            print*,'Warning *********, w(i/2) in sen_cdf_spa is zero!'
            tmp  = -10.0**10
         else
            tmp  = w(i/2)/v(i/2) - 1.0/w(i/2)**2
         endif
        Pf_w(i/2) = fai(i/2) * tmp
        if (v(i/2) .eq. 0) then
            tmp = 10.0**10
        else
            tmp = 1.0/v(i/2)**2.0
        endif
        Pf_v(i/2) = fai(i/2) * tmp

        w_t(i/2) = y/w(i/2)
        v_t(i/2) = sqrt(d2Ky(i/2))
        w_Ky(i/2) = -1.0/w(i/2)
        v_d2Ky(i/2) = ts(i/2)/(2.0* sqrt(d2Ky(i/2)))
        do j = 1, i
            ts_c(i/2,j) = -1.0* d1Ky_dc(i/2,j)/d3Ky(i/2)
            w_c(i/2,j) = w_t(i/2)*ts_c(i/2,j) + w_Ky(i/2)*
     1          (Ky_dc(i/2,j) + y*ts_c(i/2,j))
            v_c(i/2,j) = v_t(i/2)*ts_c(i/2,j) + v_d2Ky(i/2)*
     1          (d2Ky_dc(i/2,j) + d3Ky(i/2)*ts_c(i/2,j))
        enddo

        do m = 1, N
        do k = 1,  ntheta
        do j = 1, i
c     sum on j
                cs(i/2,m,k) = cs(i/2,m,k) +  (Pf_w(i/2)*w_c(i/2,j) +
     1            Pf_v(i/2)*v_c(i/2,j))*cum_sen(j,m,k)
        enddo
        enddo
        enddo

      enddo

      RETURN
      END

c------------------------------------------
      SUBROUTINE sen_cdf_spa(N,ncum,ntheta, cum, cum_sen, y, ts, cs)
c   ******  comput cdf from saddle points ******
c     input: cum, ntheta, ncum, cum_sen, y, ts   several cumulants, y : is the limitation of the performance function
c     output    :    cs  : sensitivity of cdf at y, cs(i) saved the sensitivity of cdf by 2i cumulants
c               cs(cum, N, ntheta)
c     c     ntheta is the # of distribution parameters of each variable
c           c       cum_sen(ncum,N,ntheta)
c   ****************************************

      USE FAC_INT
      USE ERFI_INT
      USE ERF_INT

      INTEGER ncum, i, j
      real*8  cum(20), ts(20),y, cs(20,20,10), w(20), v(20),
     1  Ky(20), d2Ky(20), PI , cum_sen(20,20,10), d3Ky(20),
     2  Ky_dc(20,20), d1Ky_dc(20,20), d2Ky_dc(20,20)
c       Ky_dc(ncum, derivative to ith cum)
      real*8  Pf_w(20), Pf_v(20), w_c(20,20),v_c(20,20),
     1      w_t(20), v_t(20), w_Ky(20), v_d2Ky(20), ts_c(20,20),
     2      fai(20), tmp, cdf(20)

      PI = 4.*DATAN(1.0D0)
      cs = 0.
      w = 0.
      v = 0.
      Ky = 0.
      d2Ky = 0.
      d3Ky = 0.
      Ky_dc = 0.
      d1Ky_dc = 0.
      d2Ky_dc = 0.
      Pf_w = 0.
      Pf_v = 0.
      w_c = 0.
      v_c = 0.
      w_t = 0.
      v_t=0.
      w_Ky = 0.
      v_d2Ky =0.
      ts_c = 0.
      fai(20) = 0.

      do i = 1, ncum, 1
        do j = 1, i
           Ky_dc(i,j) = 1.0/DFAC(j)*ts(i)**j
           d1Ky_dc(i,j) = 1.0/DFAC(j-1)*ts(i)**(j-1)
        enddo
        do j = 2, i
           d2Ky_dc(i,j) = 1.0/DFAC(j-2)*ts(i)**(j-2)
        enddo
C        if (i .eq. 1) then
C           Ky(i) = cum(i)/DFAC(i)*ts(i)**(i)
C           d2Ky(i) = 0.
C        else
C           do j = 1, i
C           Ky(i) = Ky(i) + cum(j)/DFAC(j)*ts(i)**(j)
C           enddo
C           do j = 2, i
C           d2Ky(i) = d2Ky(i) + cum(j)/DFAC(j-2)*ts(i)**(j-2)
C           enddo
C        endif

        if (i .eq. 1) then
           Ky(i) = cum(i)/DFAC(i)*ts(i)**(i)
           d2Ky(i) = 0.
           d3Ky(i) = 0.
        else
           do j = 1, i
           Ky(i) = Ky(i) + cum(j)/DFAC(j)*ts(i)**(j)
           enddo
           do j = 2, i
           d2Ky(i) = d2Ky(i) + cum(j)/DFAC(j-2)*ts(i)**(j-2)
           enddo

           if (i .eq. 2) then
            d3Ky(i) = 0.
           else
            do j = 3, i
            d3Ky(i) = d3Ky(i) + cum(j)/DFAC(j-3)*ts(i)**(j-3)
            enddo
           endif
        endif
        if (ts(i) .ne. 0) then
            w(i) = ts(i)/ABS(ts(i)) * (2*(ts(i)*y - Ky(i)))
     1          **0.5
        endif
        v(i) = ts(i)* d2Ky(i)**0.5
        fai(i) = 1.0/sqrt(2.0*PI)*exp(-w(i)**2.0/2.0)
        if (w(i) .eq. 0 .or. v(i) .eq. 0) then
            cdf(i) = 0.5*(1.0 + ERF(w(i)/sqrt(2.0)))
        else
            cdf(i) = 0.5*(1.0 + ERF(w(i)/sqrt(2.0))) +
     1       1.0/sqrt(2.0*PI)*exp(-w(i)**2.0/2.0)
     1        *(1.0/w(i)-1.0/v(i))
        endif
      enddo

c     **********   calculate sensitivity of cdf   ************

      do i = 1, ncum, 1
         if (w(i)*v(i) .eq. 0 ) then
            print*,'Warning ****, w(i) or v(i) in sen_cdf_spa is zero!'
            if(w(i) .eq. 0) then
            tmp  = -10.0**10
            else
            tmp = 10**10
            endif
         else
            tmp  = w(i)/v(i) - 1.0/w(i)**2
         endif
        Pf_w(i) = fai(i) * tmp
        if (v(i) .eq. 0) then
            tmp = 10.0**10
        else
            tmp = 1.0/v(i)**2.0
        endif
        Pf_v(i) = fai(i) * tmp

        w_t(i) = y/w(i)
        v_t(i) = sqrt(d2Ky(i))
        w_Ky(i) = -1.0/w(i)
        if (d2Ky(i) .ne. 0) then
        v_d2Ky(i) = ts(i)/(2.0* sqrt(d2Ky(i)))
        elseif (ts(i) .eq. 0 .and. i .ge. 3) then
           v_d2Ky(i) = 1.0/(2.0*sqrt(cum(i)/DFAC(i-3)))
        else
            v_d2Ky(i) = 0.
        endif
        do j = 1, i
            if (d2Ky(i) .eq. 0 .and. d1Ky_dc(i,j).ne.0.) then
            ts_c(i,j) = -10**10
            else if(d2Ky(i) .eq. 0 .and. d1Ky_dc(i,j).eq.0.)then
                if (i-3 .gt. 1) then
                    ts_c(i,j) = -10**10
                 else
                    ts_c(i,j) = -1.0
                 endif
            else
            ts_c(i,j) = -1.0*d1Ky_dc(i,j)/d2Ky(i)
            endif
            w_c(i,j) = w_t(i)*ts_c(i,j) + w_Ky(i)*
     1          (Ky_dc(i,j) + y*ts_c(i,j))
            v_c(i,j) = v_t(i)*ts_c(i,j) + v_d2Ky(i)*
     1          (d2Ky_dc(i,j) + d3Ky(i)*ts_c(i,j))
        enddo

        do m = 1, N
        do k = 1,  ntheta
        do j = 1, i
c     sum on j
                cs(i,m,k) = cs(i,m,k) +  (Pf_w(i)*w_c(i,j) +
     1            Pf_v(i)*v_c(i,j))*cum_sen(j,m,k)
        enddo
        enddo
        enddo

      enddo

      RETURN
      END
```